KROP moored data processing toolbox user guide

Version 1.3, August 2024

Estelle Dumont, Scottish Association for Marine Science

Table of Contents

Table of Figures

Table of Tables

1. General notes

Nomenclature in this user guide:

MM = mooring site ID code. The current list includes: BF = Billefjorden; KF= Kongsfjorden; RAF = Ramfjorden; RF = Rijpfjorden; VMF = Van Mijenfjorden.
New ID codes (2 or 3 letters long) can be created and added to this toolbox.

YY_YY = deployment start year and recovery year (e.g. 18_19)

inst = instrument type (sbe16p, sbe37, sbe56, minilog, esm1, hydrocat)

SSSS = instrument serial number

_U / _D = instrument orientation (ADCP only)

The Matlab code is managed on GitHub at:
https://github.com/EstelleDumont/KROP_mooring_toolbox

Before using the toolboxes the user must create the metadata script for each individual mooring (KROP_create_info_file_*MM_YY_YY*.m, more details in sections below). The processing can then be run without the need to edit other scripts. If code amendments are required (e.g. to add a new instrument / sensor, or to address unusual data issues) please ensure that:

- There is no hard-coding of variables or path names in the scripts.
- For additional instruments it is preferable to write new separate functions or sub-sections rather than extensively amending the existing code. Addressing one-off data or format issues should be done using cases or if statements pointing to the specific moorings / files.
- Any edits are documented.

2. Pre-processing setup

2.1. Toolboxes installation & software requirements

The following software is required:

- SeaBird SBEDataProcessing (tested with v.7.26.7)
- Matlab (tested with v. R2019b and R2024a)

The mooring processing is split into two sub-toolboxes, one for CTDs and other loggers/sensors, and one for ADCPs. Both can be downloaded from GitHub (link above).

The toolbox contains Matlab functions and libraries developed by others: CSIRO seawater Matlab library v.3.2 (P. P. Morgan, L. Pender); RDR ADCP matlab library (R. Pawlowicz); cmocean.m (C. A. Greene, K. Thynge); detect_ice.m (M. I. Wallace); cart2compass.m (A. Laurent); freezecolors (J. Iversen); keep.m (X. Yang); moving.m (A. Grinsted); mv_wind.m (A. Shaw); rgb.m (K. Jónasson); uigetdate.m (E. Tarajan);

2.2. KROP directory structure

The user should setup a root directory on the local machine, which will containing a “processing” directory and directories for each mooring.
The processing directory contains all the processing subdirectories and scripts.
The individual mooring directories will contain several subdirectories, as show in Figure 1. The user needs to create at least the **..\bin*inst*\1_raw** and **..\bin*inst*\2_converted** directories, and copy the raw instrument data in those. Other subdirectories will be created during the processing if they do not exist.

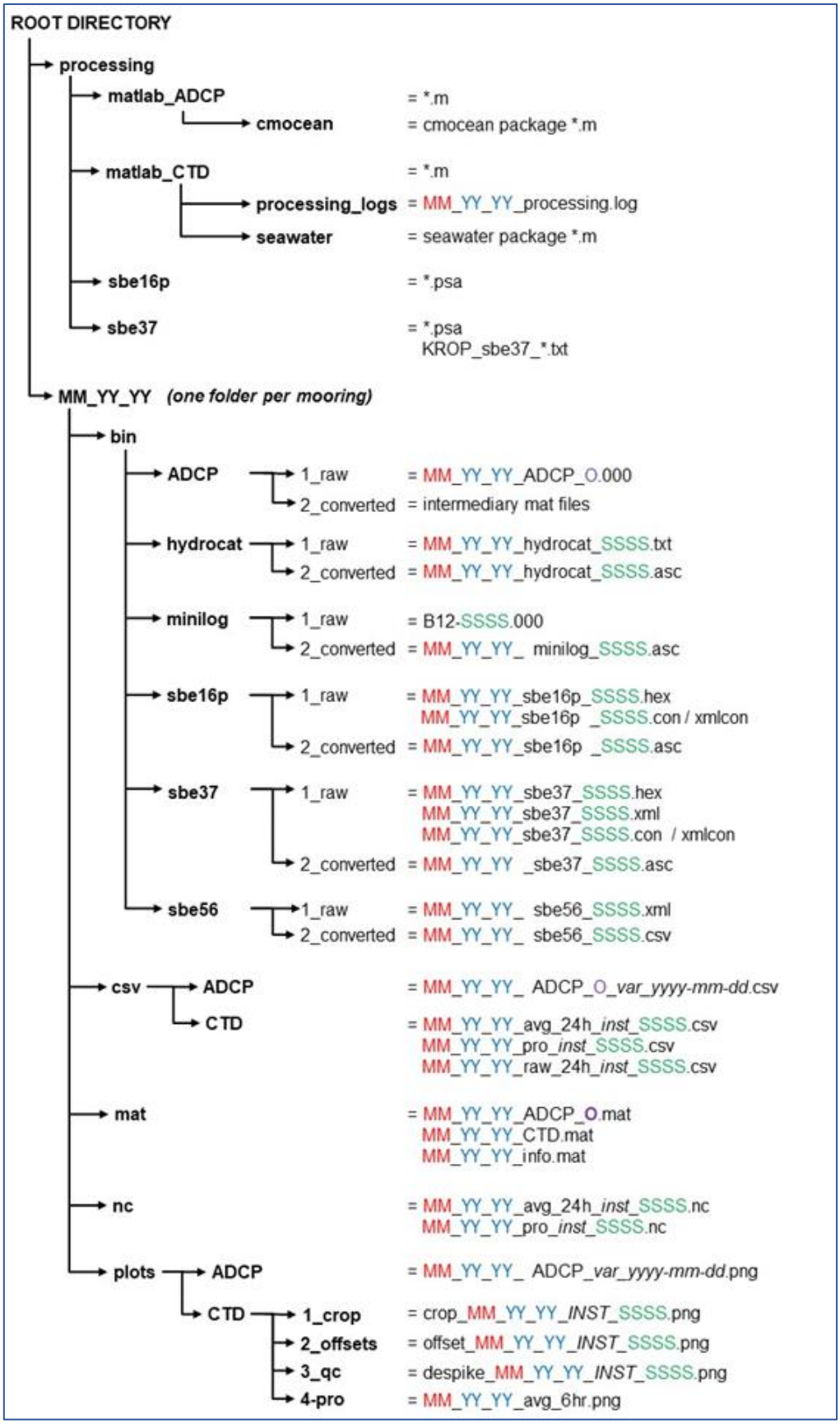

Figure 1: KROP directory structure overview

Description and origin of files in each subdirectory:

↳ **Root directory** (e.g.: C:\DATA\)
- ↳ **\processing**
  - ↳ **\matlab_ADCP**: matlab ADCP processing scripts
  - ↳ **\matlab_CTD**: matlab CTD processing scripts
    - ↳ **\processing_logs**: Matlab screen logs of the CTD processing sessions
  - ↳ **\sbe16p**: SBEDataProcessing psa files for SBE16+
  - ↳ **\sbe37**: SBEDataProcessing psa files for SBE37
- ↳ ***MM_YY_YY*** (one directory per mooring)
  - ↳ **\bin**
    - ↳ ***inst*** *(one directory per instrument type)*
      - ↳ **\1_raw**: files offloaded from instruments (.hex, .con, .000, etc)
      - ↳ **\2_converted**: raw data in standardised text format (for CTD) or intermediary mat processing files (for ADCP)
  - ↳ **\csv**
    - ↳ **\ADCP**: csv files produced at step 4.3.5.2
    - ↳ **\CTD**: csv files produced at step 3.2.10
  - ↳ **\mat**: Matlab CTD files produced at steps 3.2.1 to 3.2.7, ADCP files from step 4.3.5.1
  - ↳ **\nc**: NetCDF CTD files produced at step 3.2.11
  - ↳ **\plots**
    - ↳ **\ADCP**: ADCP contour plots produced at step 4.3.5.3
    - ↳ **\CTD**
      - ↳ **\1_crop**: plots produced at step 3.2.4
      - ↳ **\2_offsets**: plots produced at step 3.2.5
      - ↳ **\3_qc**: plots produced at step 3.2.6
      - ↳ **\4_pro**: plots produced at step 3.2.9

2.3. Metadata

See Table 1 for list of metadata information required for each mooring.
The final instrument depths along the mooring line should be established prior to starting the data post-processing. For KROP this is done via our template Excel mooring design sheet, completed with the help of the original mooring diagram, established instrument pressure offsets, and any notes recorded at deployment and recovery. This should be done by an experienced user, who is familiar with mooring hardware and pressure sensor accuracies.

The CTD toolbox allows for the input of sensor offsets, obtained from CTD intercomparison casts (pre and/or post-deployment) or other cross-calibration checks (e.g. calibration bath, or checking data against to a known in-situ value). Sensor offset determination should be done prior to the processing, and is not part of this manual or toolbox.

3. Processing: CTD data

3.1. Raw data download and formatting

Note: this step has not been described for old instruments unlikely to be used again (e.g. minilogs or ESM1) in this manual.

3.1.1. SBE16+

- Download data as a hex file in SeaTerm V2.
- Create (or amend existing) xmlcon file manually to setup sensor configuration and to add the calibration coefficients.
- Copy the two files in the directory **..\processing\sbe16p\KF** or **..\processing\sbe16p\RF**
- Rename raw hex + xmlcon files following the KROP naming convention:
    - **MM_YY_YY**_sbe16p_**SSSS**.hex
    - **MM_YY_YY**_sbe16p_**SSSS**.xmlcon
- In SBEDataProcessing run the appropriate DatCnv psa matching the mooring location and sensor configuration, e.g.:
  `DatCnv_KF_sbe16plus_ct_wetstar_satpar.psa` for SBE16s with WetStar flurorometer and Satlantic PAR sensor on KF mooring
  `DatCnv_RF_sbe16plus_ct_seapoint_satpar.psa` for SBE16s with SeaPoint flurorometer and Satlantic PAR sensor on RF mooring
  Note: new psa files might need creating for other sensor combinations.
- In SBEDataProcessing run the appropriate AsciiOut psa.
- Check the output filenames for lowercase. If needed rename them so that the mooring prefix is uppercase.
- Move the hex and xmlcon files under **..\MM_YY_YY\bin\sbe16p\1_raw**
- Move the processed cnv and asc files under **..\MM_YY_YY\bin\sbe16p\2_converted**

3.1.2. SBE19+

Note: the file format for SBE19+ is identical to SBE16+ so the same SeaBird psa files can be used for processing.

- Download data as a hex file in SeaTerm V2.
- Create (or amend existing) xmlcon file manually to setup sensor configuration and to add the calibration coefficients.
- Copy the two files in the directory **..\processing\sbe16p\KF** or **..\processing\sbe16p\RF**
- Rename raw hex + xmlcon files following the KROP naming convention:
    - **MM_YY_YY**_sbe19p_**SSSS**.hex
    - **MM_YY_YY**_sbe19p_**SSSS**.xmlcon
- In SBEDataProcessing run the appropriate DatCnv psa matching the mooring location and sensor configuration, e.g.:
  `DatCnv_KF_sbe16plus_ct_wetstar_satpar.psa` for SBE19s with WetStar flurorometer and Satlantic PAR sensor on KF mooring
  `DatCnv_RF_sbe16plus_ct_seapoint_satpar.psa` for SBE19s with SeaPoint flurorometer and Satlantic PAR sensor on RF mooring
  Note: new psa files might need creating for other sensor combinations.
- In SBEDataProcessing run the appropriate AsciiOut psa.
- Check the output filenames for lowercase. If needed rename them so that the mooring prefix is uppercase.
- Move the hex and xmlcon files under **..\MM_YY_YY\bin\sbe19p\1_raw**
- Move the processed cnv and asc files under **..\MM_YY_YY\bin\sbe19p\2_converted**

3.1.3. Hydrocat

At the point of writing this manual (2022) there was only one year of data available, recorded as screen output rather than an instrument binary file. Standardised data offload, processing and formatting will need to be established in the future.

3.1.4. SBE37

- Download data as a hex file in SeaTerm V2. The xmlcon file will be automatically uploaded as well, based on the sensor configuration and calibration coefficients set internally.
- Copy the two files in the directory **..\processing\sbe37**
- Rename raw hex + xmlcon files following the KROP naming convention:
    - **MM_YY_YY**_sbe37_**SSSS**.hex
    - **MM_YY_YY**_sbe37_**SSSS**.xmlcon
- Open Run window (Ctrl+R) and run the appropriate batch file:

    `KROP_sbe37_KF_ptc` is for SBE37s with pressure sensor on KF mooring
    `KROP_sbe37_KF_tc` is for SBE37s without pressure sensor on KF mooring
    `KROP_sbe37_RF_ptc` is for SBE37s with pressure sensor on RF mooring
    `KROP_sbe37_RF_tc` is for SBE37s without pressure sensor on RF mooring

    Example command lines:

```
sbebatch  C:\ KROP\sbe37\KROP_sbe37_KF_ptc.txt  KF_16_17_sbe37_1234
C:\ KROP\processing\sbe37
sbebatch  C:\ KROP\sbe37\KROP_sbe37_RF_ptc.txt  RF_16_17_sbe37_1234
C:\ KROP\processing\sbe37
sbebatch  C:\ KROP\sbe37\KROP_sbe37_KF_tc.txt  KF_16_17_sbe37_1234
C:\ KROP\processing\sbe37
sbebatch  C:\ KROP\sbe37\KROP_sbe37_RF_tc.txt  RF_16_17_sbe37_1234
C:\ KROP\processing\sbe37
```

- The batch processing will export data as cnv and asc but might make the filenames all lowercase. Rename them so that the mooring prefix is uppercase.
- Move the hex and xmlcon files under **..\MM_YY_YY\bin\sbe37\1_raw**
- Move the processed cnv and asc files under **..\MM_YY_YY\bin\sbe37\2_converted**

3.1.5. SBE56

- Download data as a xml file in SeaTerm V2.
- Export data to cnv (Tools → Convert .XML data file).

Note for all SBE CTDs: if hex or xml files are not available, the user should reformat the raw files to match the toolbox input requirements described in section 3.2.3.

3.2. Matlab toolbox processing

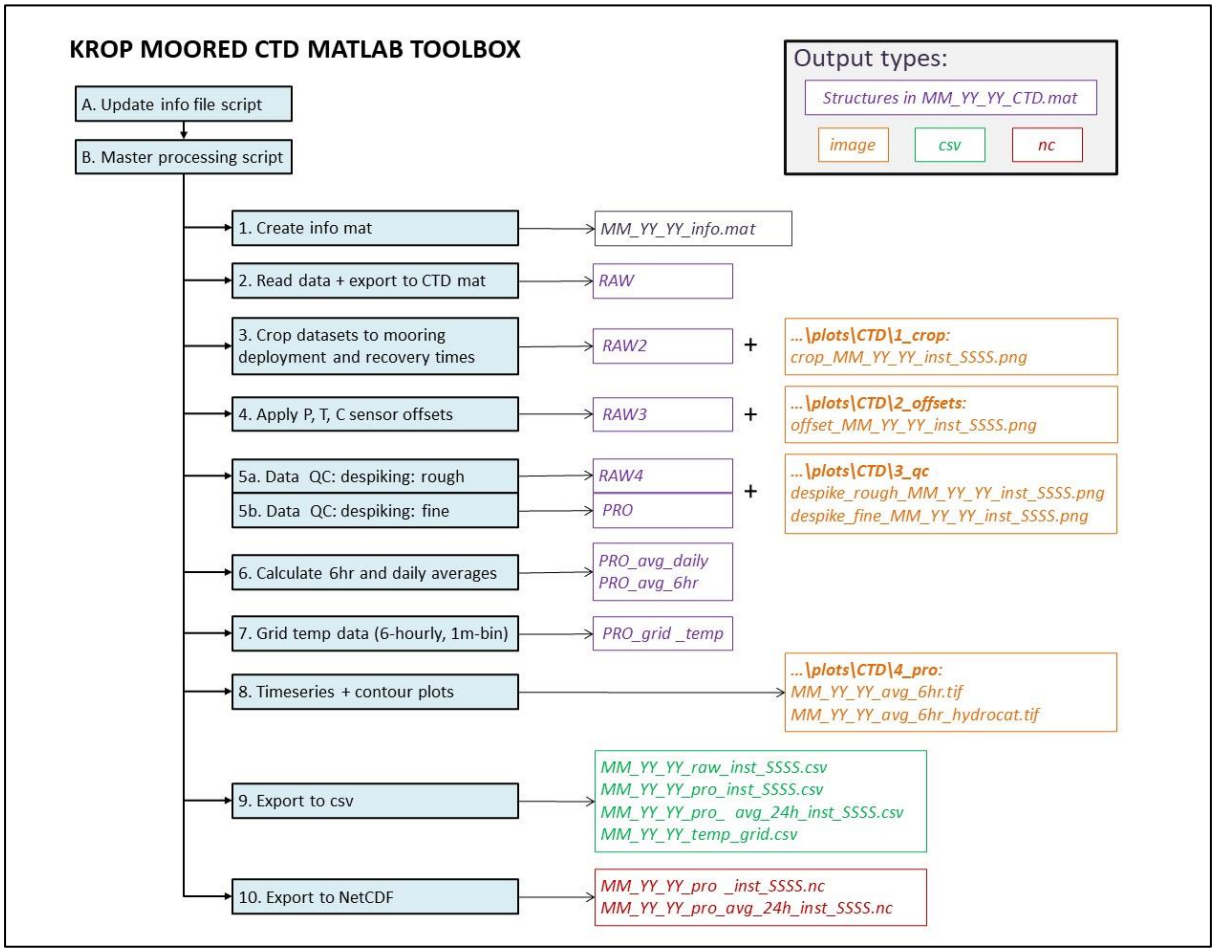

Figure 2: KROP moored CTD toolbox processing overview

3.2.1. Create metadata file for each mooring

Script: KROP_create_info_file_*MM_YY_YY*.m
Sub-routines called: N/A
Input(s): N/A
Output(s): *N/A*

This script must be created/edited for each mooring before running the processing. The variables to define and formats are listed in the table below.
At this step the script needs to be edited only, not run.

Note: if two instruments are located at the same depth it is necessary to offset them slightly (by 0.1m) in this file in order for the gridding script to work properly.

Table 1: List of metadata in mooring info file

Name	Manda-tory?	Definition	Format / type	Example(s)		
d_root	Y	Root directory path on processing machine	string	'C:\DATA'		
mooring_id	Y	Mooring identifier in format MM(M)_YY_YY	string	'RF_18_20'		
start_date	Y	Mooring deployment date and UTC time (first round hour after the mooring has settled on the seabed)	dd-mmm-yyyy HH:MM:SS	datenum('14-Aug-2018 16:00:00')		
end_date	Y	Mooring deployment date and UTC time (last round hour before the mooring has been released from the seabed)	dd-mmm-yyyy HH:MM:SS	datenum('09-Sep-2020 06:00:00')		
moor_lat_deg	Y	Mooring latitude degrees	float	80		
moor_lat_min	Y	Mooring latitude decimal minutes	float	17.710		
moor_lon_deg	Y	Mooring longitude degrees	float	22		
moor_lon_min	Y	Mooring longitude decimal minutes	float	17.943		
mooring_depth	Y	Mooring / seabed depth in metres	float	233		
sbe16p_num	Y	Number of SBE16+	float	0	1	2
sbe16p_sn	YES if sbe16p_num >0	SBE16+ serial number(s)	string(s) in cell array	' '	{'6066'}	{'6066' '6067'}
sbe16p_depth		SBE16+ nominal depth in metres, in the same order as S/N array	string(s) in cell array	' '	{'40'}	{'20' '40'}
sbe16p_p		SBE16+ pressure sensor indicator(s) 0 = *not used*, 1 = present, 2= *not used*, 3 = faulty	float	' '	1	[1;1]
off_sbe16p_t		SBE16+ temperature offset(s) to apply NaN if offset has not been determined	float	' '	0.01	[0.01;0]
off_sbe16p_c		SBE16+ conductivity offset(s) to apply NaN if offset has not been determined	float	' '	0	[0;-0.02]
sbe16p_p_cal_date		SBE16+ pressure sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '30-Jul-14'}		
sbe16p_t_cal_date		SBE16+ temperature sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '30-Jul-14'}		
sbe16p_c_cal_date		SBE16+ conductivity sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '30-Jul-14'}		
sbe16p_flc		SBE16+ fluorometer indicator(s): 0 = absent, 1 = present	float	[1;1]		
sbe16p_flc_sn		SBE16+ fluorometer serial number(s)	string(s) in cell array	{'3585' '3586'};		
sbe16p_flc_type		SBE16+ fluorometer model(s)	string(s) in cell array	{'Seapoint' 'WetLabs WETstar'}		
sbe16p_flc_cal_date		SBE16+ fluorometer calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '30-Jul-14'}		
sbe16p_par		SBE16+ PAR sensor indicator(s) 0 = absent, 1 = present	float	[1;1]		
sbe16p_par_sn		SBE16+ PAR sensor serial number(s)	string(s) in cell array	{'1010' '1011'}		
sbe16p_par_type		SBE16+ PAR sensor model(s)	string(s) in cell array	{'Satlantic' 'Biospherical'}		

Name	Manda-tory?	Definition	Format / type	Example(s)		
sbe16p_par_cal_date		SBE16+ PAR sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '30-Jul-14'}		
sbe16p_tur		SBE16+ turbidity sensor indicator(s) 0 = absent, 1 = present	float	[1;1]		
sbe16p_tur_type		SBE16+ turbidity sensor serial number(s)	string(s) in cell array	{'1234' '1235'}		
sbe16p_tur_sn		SBE16+ turbidity sensor model(s)	string(s) in cell array	{'Seapoint' 'Seapoint'}		
sbe16p_tur_cal_date		SBE16+ turbidity sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '29-Jul-2014'}		
sbe19p_num	Y	Number of SBE19+	float	0	1	2
sbe19p_sn	YES if sbe19p_num >0	SBE19+ serial number(s)	string(s) in cell array	' '	{'6824'}	{'6824' '6825'}
sbe19p_depth		SBE19+ nominal depth in metres, in the same order as S/N array	string(s) in cell array	' '	{'30'}	{'20' '40'}
sbe19p_p		SBE19+ pressure sensor indicator(s) 0 = *not used*, 1 = present, 2= *not used*, 3 = faulty	float	' '	1	[1;1]
off_sbe19p_t		SBE19+ temperature offset(s) to apply NaN if offset has not been determined	float	' '	0.01	[0.01;0]
off_sbe19p_c		SBE19+ conductivity offset(s) to apply NaN if offset has not been determined	float	' '	0	[0;-0.02]
sbe19p_p_cal_date		SBE19+ pressure sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '30-Jul-14'}		
sbe19p_t_cal_date		SBE19+ temperature sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '30-Jul-14'}		
sbe19p_c_cal_date		SBE19+ conductivity sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '30-Jul-14'}		
sbe19p_flc		SBE19+ fluorometer indicator(s): 0 = absent, 1 = present	float	[1;1]		
sbe19p_flc_sn		SBE19+ fluorometer serial number(s)	string(s) in cell array	{'3585' '3586'};		
sbe19p_flc_type		SBE19+ fluorometer model(s)	string(s) in cell array	{'Seapoint' 'WetLabs WETstar'}		
sbe19p_flc_cal_date		SBE19+ fluorometer calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '30-Jul-14'}		
sbe19p_par		SBE19+ PAR sensor indicator(s) 0 = absent, 1 = present	float	[1;1]		
sbe19p_par_sn		SBE19+ PAR sensor serial number(s)	string(s) in cell array	{'1010' '1011'}		
sbe19p_par_type		SBE19+ PAR sensor model(s)	string(s) in cell array	{'Satlantic' 'Biospherical'}		
sbe19p_par_cal_date		SBE19+ PAR sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '30-Jul-14'}		
sbe19p_tur		SBE19+ turbidity sensor indicator(s) 0 = absent, 1 = present	float	[1;1]		
sbe19p_tur_type		SBE19+ turbidity sensor serial number(s)	string(s) in cell array	{'1234' '1235'}		
sbe19p_tur_sn		SBE19+ turbidity sensor model(s)	string(s) in cell array	{'Seapoint' 'Seapoint'}		
sbe19p_tur_cal_date		SBE19+ turbidity sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '29-Jul-2014'}		

Name	Manda-tory?	Definition	Format / type	Example(s)		
hcat_num	Y	Number of Hydrocat	float	0	1	2
hcat_sn	YES if hcat_num >0	Hydrocat serial number(s)	string(s) in cell array	' '	{'30236'}	{'1234' '1235'}
hcat_depth		Hydrocat nominal depth in metres, in the same order as S/N array	string(s) in cell array	' '	{'29'}	{'20' '40'}
hcat_p		Hydrocat pressure sensor indicator(s) 0 = *not used*, 1 = present, 2= *not used*, 3 = faulty	float	' '	1	[1;1]
off_hcat_t		Hydrocat temperature offset(s) to apply NaN if offset has not been determined	float	' '	0.01	[0.01;0]
off_hcat_c		Hydrocat conductivity offset(s) to apply NaN if offset has not been determined	float	' '	0	[0;-0.02]
hcat_p_cal_date		Hydrocat pressure sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '30-Jul-14'}		
hcat_t_cal_date		Hydrocat temperature sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '30-Jul-14'}		
hcat_c_cal_date		Hydrocat conductivity sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2014' '30-Jul-14'}		
hcat_oxy		Hydrocat oxygen sensor indicator(s) 0 = absent, 1 = present	float	1		
hcat_oxy_sn		Hydrocat oxygen sensor serial number(s)	string(s) in cell array	{'1781'}		
hcat_oxy_type		Hydrocat oxygen sensor model(s)	string(s) in cell array	{'SBE63'}		
hcat_oxy_cal_date		Hydrocat oxygen sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'04-Oct-2017'}		
hcat_ph		Hydrocat pH sensor indicator(s) 0 = absent, 1 = present	float	1		
hcat_ph_sn		Hydrocat pH sensor serial number(s)	string(s) in cell array	{'114'}		
hcat_ph_type		Hydrocat pH sensor model(s)	string(s) in cell array	{'HydroCAT-pH'}		
hcat_ph_cal_date		Hydrocat pH sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'06-Nov-2017'}		
hcat_flc		Hydrocat fluorometer indicator(s): 0 = absent, 1 = present	float	1		
hcat_flc_sn		Hydrocat fluorometer serial number(s)	string(s) in cell array	{'108'}		
hcat_flc_type		Hydrocat fluorometer model(s)	string(s) in cell array	{'WetLabs HCO'}		
hcat_flc_cal_date		Hydrocat fluorometer calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'12-Oct-2017'}		
hcat_tur		Hydrocat turbidity sensor indicator(s) 0 = absent, 1 = present	float	1		
hcat_tur_type		Hydrocat turbidity sensor serial number(s)	string(s) in cell array	{'108'}		
hcat_tur_sn		Hydrocat turbidity sensor model(s)	string(s) in cell array	{'WetLabs HCO'}		
hcat_tur_cal_date		Hydrocat turbidity sensor calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'12-Oct-2017'}		

Name	Manda-tory?	Definition	Format / type	Example(s)
sbe37_num	Y	Number of SBE37	float	2
sbe37_sn	YES if sbe37_num >0	SBE37 serial number(s)	string(s) in cell array	{'7248' '9114'}
sbe37_depth		SBE37 nominal depth in metres, in the same order as S/N array	string(s) in cell array	{'34' '224'}
sbe37_p		SBE37 pressure sensor indicator(s) 0 = absent, 1 = present, 2= special formatting, 3 = bad data	float	[0;1]
off_sbe37_t		SBE37 temperature offset(s) to apply NaN if offset has not been determined	float	[0;0]
off_sbe37_c		SBE37 conductivity offset(s) to apply NaN if offset has not been determined	float	[0;0]
sbe37_cal_date		SBE37 calibration date(s) – same for P, T and C	dd-mmm-yyyy HH:MM:SS	{'29-Jul-2015' '26-Jun-2011'}
sbe56_num	Y	Number of SBE56	float	4
sbe56_sn	YES if sbe56_num >0	SBE56 serial number(s)	string(s) in cell array	{'2662' '2663' '2664' '2665'}
sbe56_depth		SBE56 nominal depth in metres, in the same order as S/N array	string(s) in cell array	{ '30' '40' '50' '60'}
off_sbe56_t		SBE56 temperature offset(s) to apply NaN if offset has not been determined	float	[0;0;0;0]
sbe56_cal_date		SBE56 calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'31-Jul-2013' '31-Jul-2013' '31-Jul-2013' '31-Jul-2013'}
ml_num	Y	Number of Vemco minilogs	float	3
ml_sn	YES if ml_num >0	Minilogs serial number(s)	string(s) in cell array	{'5231' '5230' '5232'}
ml_depth		Minilogs nominal depth in metres, in the same order as S/N array	string(s) in cell array	{'43.5' '53.5' '63.5'}
off_ml_t		Minilogs temperature offset(s) to apply NaN if offset has not been determined	float	[0.03;0.02;0.05]
ml_cal_date		Minilogs calibration date(s) (can be left as 'unknown')	dd-mmm-yyyy HH:MM:SS	{'unknown' 'unknown' 'unknown'}
ml_p		Minilogs pressure sensor indicator(s)	float	[0;0;1]
esm1_num	N	Number of ESM-1 loggers	float	1
esm1_sn	Y if esm1_num >0	ESM-1 serial number(s)	string(s) in cell array	{'2341'}
esm1_depth		ESM-1 nominal depth in metres, in the same order as S/N array	string(s) in cell array	{'23.5'}
esm1_flc		ESM-1 fluorometer indicator(s): 0 = absent, 1 = present	float	[1]
esm1_flc_sn		ESM-1 fluorometer serial number(s)	string(s) in cell array	{'2353'}
esm1_flc_type		ESM-1 fluorometer model(s)	string(s) in cell array	{'Seapoint'}
esm1_flc_cal_date		ESM-1 fluorometer calibration date(s)	dd-mmm-yyyy HH:MM:SS	{'08-Feb-2001'};

Name	Manda-tory?	Definition	Format / type	Example(s)
esm1_flc_bits		ESM-1 logger I/O bits resolution	float	16
esm1_flc_range		ESM-1 fluorometer range in µg/l (Options: gain 1X=150 µg/l, 3X=50, 10X=15, 30X=5)	float	50
ga_inst	Y	NetCDF file global attribute: institutions who have generated the dataset	string	'UiT University of Tromso, SAMS Scottish Association for Marine Science'
ga_proj	Y	NetCDF file global attribute: project name	string	'KROP'
ga_array	Y	NetCDF file global attribute: mooring array	string	'KROP'
ga_cont	Y	NetCDF file global attribute: contributors list	string	'Jorgen Berge; Finlo Cottier; Estelle Dumont; Daniel Vogedes'
ga_cmt	Y	NetCDF file global attribute: general comment, applicable to all datafiles for this mooring (NOT individual instruments or variables)	string	'Data looking acceptable'
ga_cmt_*SSSS*	N	NetCDF file global attribute: general comment for an individual instrument	string	'Data looking acceptable. Instrument stopped recording part-way through deployment.'

3.2.2. Call master processing script

Script: KROP_master_process.m
Sub-routines called: KROP_create_info_file_MM_YY_YY.m & all others
Input(s): on-screen user inputs
Output(s): MM_YY_YY_info.mat, MM_YY_YY_processing_log.txt

Once the metadata script has been setup, the full processing is started from the master script which will in turn call the rest of the processing routines.
In Matlab, the script is called with the command:

```
KROP_master_process
```

The script will ask the user for:

- Mooring ID: this should be in the format **MM_YY_YY** (e.g. RF_18_20) or **MMM_YY_YY**
- Processor: name of the person processing this mooring's dataset

This script will create the metadata mat file (MM_YY_YY_info.mat), and create the required sub-directories in the main mooring directory (except the \bin one, which should be created by the user beforehand).

3.2.3. Read data

Script: KROP_read_data.m
Sub-routines called: KROP_read_sbe16p.m, KROP_read_sbe19p.m, KROP_read_hcat.m, KROP_read_sbe37.m, KROP_read_sbe56.m, KROP_read_minilog.m, KROP_read_esm1.m
Input(s): MM_YY_YY_info.mat, instruments raw data in standardised text format
Output(s): MM_YY_YY_CTD.mat ('RAW' structure)

For all instruments a nominal depth field will be added (using the values from the info file) and the Matlab time.
If the CTD mat file already exists (in case the user had started the processing previously) the script will ask if the existing file can be used or if the raw data should be read again.

3.2.3.1. SBE16+

The script will determine the file format using the sensors configuration set in the info file. The default input file formats are comma-separated ascii files produced by the SBEDataProcessing software containing one header line and the following variables (in this order):

Table 2: input file format – SBE16+

Variable	Mandatory	Description	Units / format
Scan	Yes	Instrument scan number	n/a
TimeJV2	Yes	Julian day	n/a
PrdM	Yes	Pressure	db
DepSM	Yes	Depth	m
TV290C	Yes	Temperature (ITS-90)	°C
C0mS/cm	Yes	Conductivity	mS/cm
Sal00	Yes	Practical salinity	psu
Sigma-é00	Yes	Density sigma-theta σ_θ	kg/m^3 - 1000
V[0]	No	Fluorometer voltage channel	volts
Fluorescence	No	Chlorophyll-a concentration	µg/l
V[0]	No	PAR sensor voltage channel	volts
PAR	No	PAR	µmol photons/m^2/s or µEinsteins/m^2/s
V[0]	No	Turbidity voltage channel	volts

Variable	Mandatory	Description	Units / format
Turbidity	No	Turbidity	FTU
Flag	Yes	SBE processing flag	n/a

If the pressure sensor is set as faulty (sbe16p_p=3) the depth field will be replaced by the nominal deployment depth from the mooring design sheet, and the associated variables re-calculated (pressure, salinity, sigma-theta) using the Matlab seawater routines.
Cases of absent pressure sensor (sbe16p_p=0) or special file formatting (sbe16p_p=2) have not been encountered in the KROP datasets so far and therefore have not been coded, but this could be added in this script if required.

3.2.3.2. SBE19+

The script will determine the file format using the sensors configuration set in the info file. The default input file formats are comma-separated ascii files produced by the SBEDataProcessing software containing one header line and the following variables (in this order):

Table 3: input file format – SBE19+

Variable	Mandatory	Description	Units / format
Scan	Yes	Instrument scan number	n/a
TimeJV2	Yes	Julian day	n/a
PrdM	Yes	Pressure	db
DepSM	Yes	Depth	m
TV290C	Yes	Temperature (ITS-90)	°C
C0mS/cm	Yes	Conductivity	mS/cm
Sal00	Yes	Practical salinity	psu
Sigma-é00	Yes	Density sigma-theta σ_θ	kg/m^3 - 1000
V[0]	No	Fluorometer voltage channel	volts
Fluorescence	No	Chlorophyll-a concentration	µg/l
V[0]	No	PAR sensor voltage channel	volts
PAR	No	PAR	µmol photons/m^2/s or µEinsteins / m^2 / s
V[0]	No	Turbidity voltage channel	volts
Turbidity	No	Turbidity	FTU
Flag	Yes	SBE processing flag	n/a

If the pressure sensor is set as faulty (sbe19p_p=3) the depth field will be replaced by the nominal deployment depth from the mooring design sheet, and the associated variables re-calculated (pressure, salinity, sigma-theta) using the Matlab seawater routines.
Cases of absent pressure sensor (sbe19p_p=0) or special file formatting (sbe19p_p=2) have not been encountered in the KROP datasets so far and therefore have not been coded, but this could be added in this script if required.

3.2.3.3. Hydrocat

The script will determine the file format using the sensors configuration set in the info file. The current default input file format is the terminal screen capture of the data record (obtained with the command 'getsamples'), plus one line of header added manually.
In the future, since different instruments might have different output formats, it would be useful to download and export the data in a standardised text format.
The current variables order the script will read is:

Table 4: input file format – Hydrocat

Variable	Mandatory	Description	Units / format
Header	Yes	Instrument default line start	n/a
Temp	Yes	Temperature (ITS-90)	°C
Cond	Yes	Conductivity	µS/cm
Press	Yes	Pressure	db
Oxy	Yes	Dissolved oxygen	mg/L
pH	Yes	pH	pH
Chl	Yes	Chlorophyll-a concentration	µg/l
Turb	Yes	Turbidity	NTU
Chl_std	Yes	Chl standard deviation	µg/l
Turb_std	Yes	Turb standard deviation	NTU
Salinity	Yes	Practical salinity	psu
SpecCond	Yes	Specific conductivity	µS/cm
OxySat	Yes	Oxygen saturation	%
Date	Yes	Date	dd mmm yyyy
Time	Yes	Time	HH:MM:SS
Flag	Yes	SBE processing flag	n/a

Additionally, the script:

- Adds a scan number to each data point
- Converts conductivity from µS/cm to mS/cm
- Calculates depth and sigma-theta

If the pressure sensor is set as faulty (hcat_p=3) the depth field will be replaced by the nominal deployment depth from the mooring design sheet, and the associated variables re-calculated (pressure, salinity, sigma-theta) using the Matlab seawater routines.
Cases of absent pressure sensor (hcat_p=0) or special file formatting (hcat_p=2) have not been coded, but this could be added in this script if required.

3.2.3.4. SBE37

The script will determine the file format using the sensors configuration from the info file. The default input file formats are comma-separated ascii files produced by the SBEDataProcessing software containing one header line and the following variables (in this order):

Table 5: input file format – SBE37

Variable	Mandatory	Description	Units / format
Scan	Yes	Instrument scan number	n/a
TimeJV2	Yes	Julian day	n/a
PrdM	Yes	Pressure	db
DepSM	Yes	Depth	m
TV290C	Yes	Temperature (ITS-90)	°C
C0mS/cm	Yes	Conductivity	mS/cm
Sal00	Yes	Practical salinity	psu
Sigma-é00	Yes	Density sigma-theta σ_θ	kg/m^3 - 1000
Flag	Yes	SBE processing flag	n/a

Note: during the course of the data conversion process two different file formats were exported, switching the order of the Pressure and Depth variables; the script will check the header line to determine which format is being used.
If there is no pressure sensor (sbe37_p=0) the depth field will be replaced by the nominal deployment depth from the mooring design sheet, and the associated variables re-calculated (pressure, salinity,

sigma-theta) using the Matlab seawater routines.
If there is a pressure sensor (sbe37_p=1) the script will check if derived variables have been calculated (depth, salinity, sigma-theta) or if they are set to NaNs. This happened for older instruments where no hex file was available, in this case the raw instrument files could not be processed in the SBE software and had to be manually reformatted. If the variables are missing the script will re-calculate them using the Matlab seawater routines.
In some (historical) instances the only datafile available was from a screen output of the instrument memory in the SeaBird software ('dd' command). This is indicated in the info file as sbe37_p=2. The script will read this file format, convert the conductivity from S/m to mS/cm, and re-calculate any missing variables (pressure, sigma-theta) using the Matlab seawater routines.
If the pressure sensor is set as faulty (sbe37_p=3) the depth field will be replaced by the nominal deployment depth from the mooring design sheet, and the associated variables re-calculated (pressure, salinity, sigma-theta) using the Matlab seawater routines.

3.2.3.5. SBE56

SBE56 loggers input files are csv files produced by the SBE software SeaTermV2 (default format), which will contain the following variables (in this order):

Table 6: input file format – SBE56

Variable	Mandatory	Description	Units / format
Date	Yes	Date	dd/mm/yyyy *or* yyyy-mm-dd *or* dd-mm-yyyy
Time	Yes	Time UTC	hh:mm:ss
Temperature	Yes	Temperature (ITS-90)	°C

3.2.3.6. Minilog

The minilogs input files are the comma-separated ascii files exported from the Vemco software. The script will determine the file format using 1) the pressure sensor indicator set in the info file and 2) the year of deployment (different formats across the software versions). The raw formats handled by the script are:

Table 7: input file format – minilog

Deployment year	Header lines	Format
Up to 2007	7	DD-MM-YYYY, hh:mm:ss, temperature, depth (optional)
2008-2009	7	YYYY-MM-DD, hh:mm:ss, temperature, depth (optional)
From 2010	8	DD, MM, YYYY, hh, mm, ss, temperature, temperature (A/D)

3.2.3.7. ESM1

An ESM1 logger has been deployed on very early KROP moorings, a script has been written to read and convert the data from raw bits to chl-a concentration. This instrument is now obsolete and not expected to be used again. The input file format used was:

Table 8: input file format – ESM1

Variable	Mandatory	Description	Units / format
Julian Day	Yes	Julian day	n/a
Fluo raw	Yes	Fluorescence sensor readings	A/D counts
Press raw	Yes	Pressure sensor readings	A/D counts
Temp raw	Yes	Temperature sensor readings	A/D counts
Volts	Yes	Logger battery voltage	volts
Command	Yes	Logger status	n/a
Fluo	Yes	Chlorophyll-a concentration	µg/l

3.2.4. Crop data to mooring deployment and recovery times

Script: KROP_crop.m
Sub-routines called: none
Input(s): MM_YY_YY_info.mat, MM_YY_YY_CTD.mat ('RAW' structure)
Output(s): MM_YY_YY_CTD.mat ('RAW2' structure)

The script will read the moorings deployment and recovery time from the info file and crop each instrument dataset to the correct date range. It will plot the results (example below) and the user will have the option to accept or reject the selection (the processing will be halted in this case).

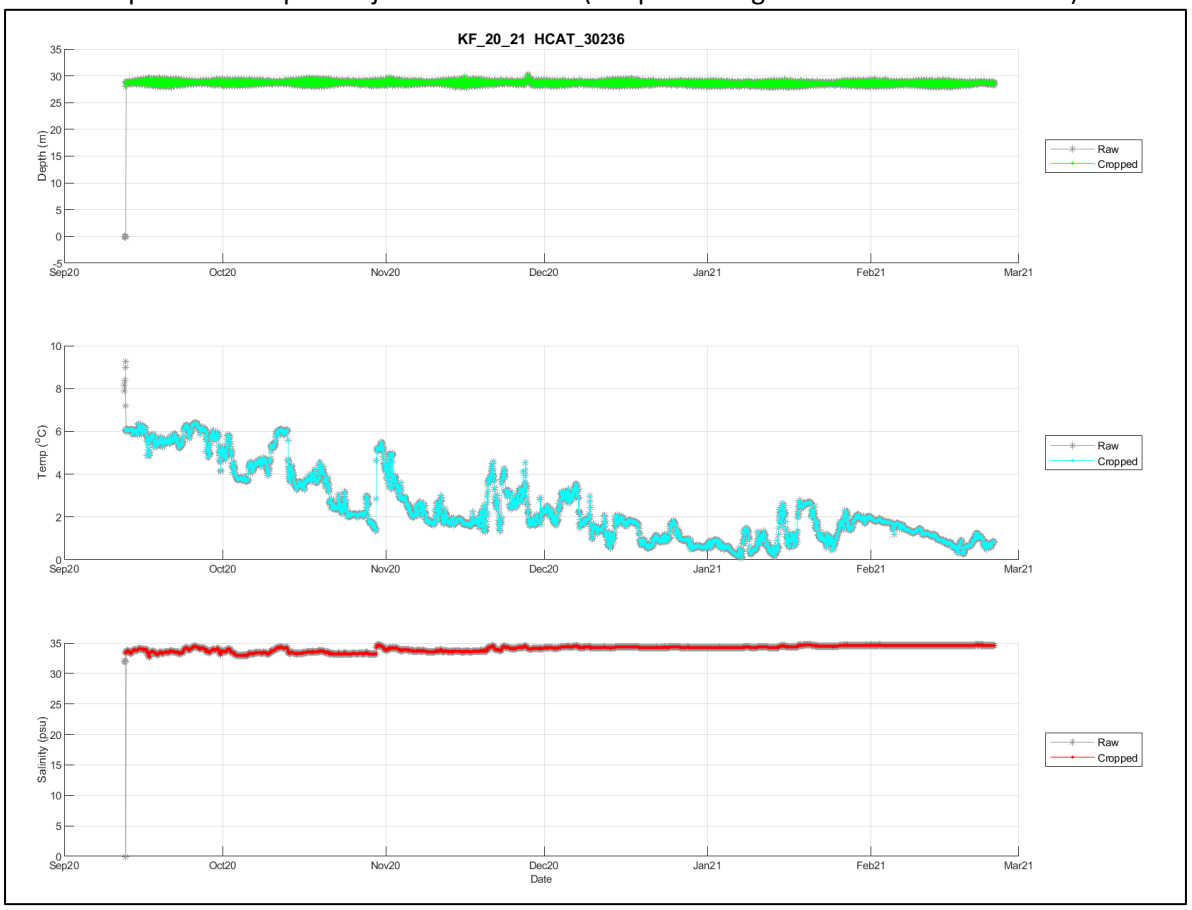

Figure 3: example of plot for step 3 – cropping data

3.2.5. Apply sensor offsets

Script: KROP_apply_offsets.m
Sub-routines called: none
Input(s): MM_YY_YY_info.mat, MM_YY_YY_CTD.mat ('RAW2' structure)
Output(s): MM_YY_YY_CTD.mat ('RAW3' structure)

For each instrument the script will read the temperature and conductivity offsets from the info file, apply those to the raw values and plot the results. The offsets can be determined from:

- CTD inter-comparison casts (with 10+ minutes bottles stops)
- calibration baths
- adjustment to a known in-situ value (e.g. winter bottom temperature minima in Rijpfjorden)

The offsets specified in the info file should be the offset to add to the raw values. Salinity and sigma-theta will be recalculated from the corrected temperature and conductivity data using the Matlab seawater routines.

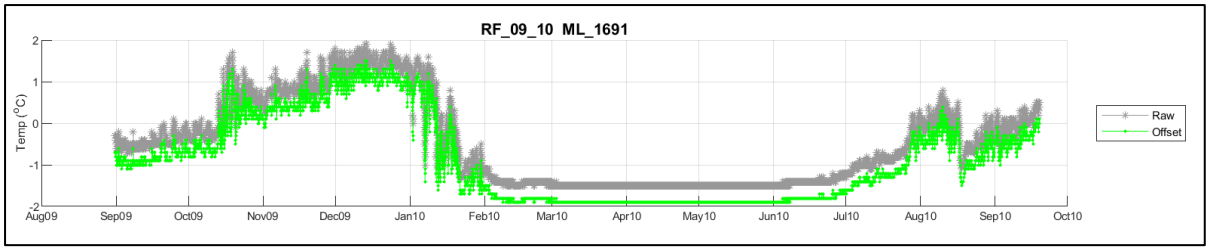

Figure 4: example of plot for step 4 – applying CT offsets

3.2.6. Despike data

Script: KROP_qc_rough.m, KROP_qc_fine.m
Sub-routines called: KROP_flag_bad_data.m, mv_wind.m
Input(s): MM_YY_YY_info.mat, MM_YY_YY_CTD.mat ('RAW3' structure)
Output(s): MM_YY_YY_CTD.mat ('RAW4' and 'PRO' structure)

This script performs an automated despiking of the main variables for each instrument (temperature, conductivity and salinity). No QC is performed on fluorescence, PAR or other optional variables.

The despiking is done in two stages: the first "rough" pass is to eliminate large spikes, the second "fine" pass to identify smaller spikes / noise.

For each variable the script will calculate median values based on a set moving window (expressed in hours), flag points that fall beyond the set tolerance (expressed in standard deviations from the mean), and plot the results. The user will then have the option to either accept the selection, or to set a new window size and/or tolerance(s). This can be repeated until satisfactory values have been established.
The starting values are:

Window length = 4 hours
Tolerance temperature = 2 standard deviations
Tolerance conductivity = 2 standard deviations
Tolerance salinity = 0.5 standard deviation

Those values have been found to work quite well across the KROP datasets to pick up large outliers, for smaller spikes the user might need to extend the averaging window (to 8 or 12h, sometimes 24h) and reduce the tolerance (~0.5 or 1 std for temperature and conductivity, 0.25 for salinity usually manage to capture the remaining smaller spikes during the second pass).
It can sometimes be difficult to determine what is real variation in the water column and what is bad data, particularly for instruments located near the surface which can suffer from noisier readings. The despiking of the KROP datasets has been done conservatively, i.e. if unsure if a datapoint is good or bad: leave it in the dataset.
Rough and fine despiking settings will be saved in the final mat and netcdf files.

When a temperature datapoint is flagged as bad the value will be changed to NaN, as well as the salinity and sigma-theta values.
When a conductivity datapoint is flagged as bad the value will be changed to NaN, as well as the salinity and sigma-theta values.
When a salinity datapoint is flagged as bad the value will be changed to NaN, as well as the sigma-theta value.

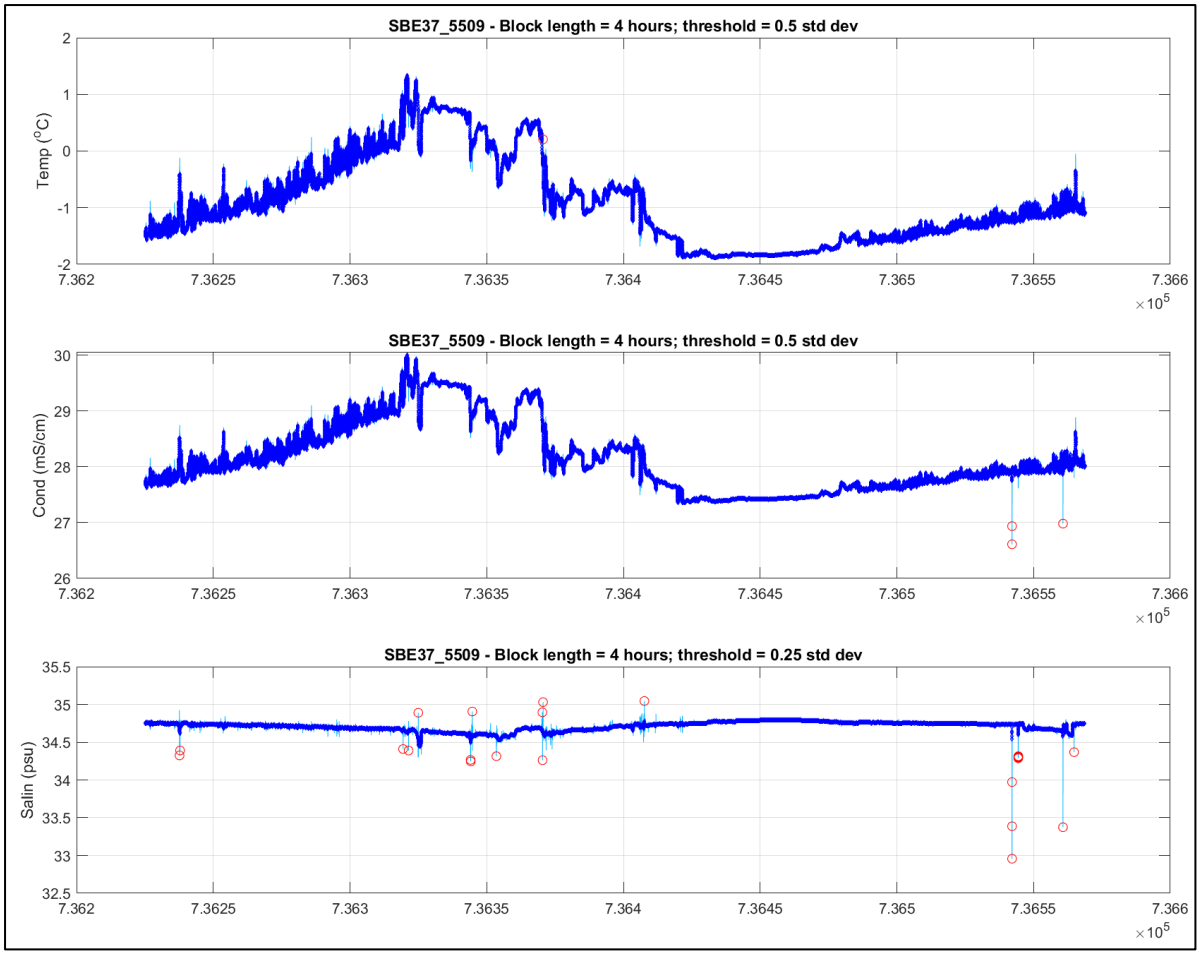

Figure 5: example of plot for step 5 – despiking. Cyan points show the raw dataset, dark blue the calculated median, and red circles indicate the flagged datapoints.

3.2.7. Average data

Script: KROP_avg_daily.m, KROP_avg_6hr.m
Sub-routines called: none
Input(s): MM_YY_YY_info.mat, MM_YY_YY_CTD.mat (‘PRO’ structure)
Output(s): MM_YY_YY_CTD.mat ('PRO_avg_daily' and 'PRO_avg_6hr' structures)

For all variables of each instrument, the script averages all datapoints falling within +/- half the averaging time:

- for daily files the data is averaged around 12:00 UTC (i.e. all readings collected between 00:00 UTC and 23:59:59 UTC)
- for 6-hourly files the data is averaged around 00:00, 06:00, 12:00 and 18:00 UTC.

The mtime and Julian days values in the structures are the actual averaged times for each interval, which might be slightly different from the nominal average time.

3.2.8. Grid temperature data

Script: KROP_grid_temp.m
Sub-routines called: none
Input(s): MM_YY_YY_info.mat, MM_YY_YY_CTD.mat ('PRO_avg_6hr' structure)
Output(s): MM_YY_YY_CTD.mat ('PRO_grid_temp' structure)

This script collates the temperature data from all instruments installed on the mooring, using the 6-hourly averaged datasets. It then linearly interpolates the data vertically every 1m at each timestep.

3.2.9. Plot final data

Script: KROP_plot_mooring.m
Sub-routines called: none
Input(s): MM_YY_YY_info.mat, MM_YY_YY_CTD.mat (PRO_avg_6hr + PRO_grid_temp structures)
Output(s): MM_YY_YY_avg_6hr.tif, *optional:* MM_YY_YY_avg_6hr_hydrocat.tif

This script produces plots of the final processed data (6hr-averaged):

- Time-series plots of:
  - Fluorescence: up to 2 SBE16+, 2 SBE19+, 2 Hydrocat, 2 ESM-1, 6 SBE37
  - PAR: up to 2 SBE16+, 2 SBE19+, 2 Hydrocat, 6 SBE37
  - Salinity: up to 2 SBE16+, 2 SBE19+, 2 Hydrocats,6 SBE37
- Contour plot of gridded temperature

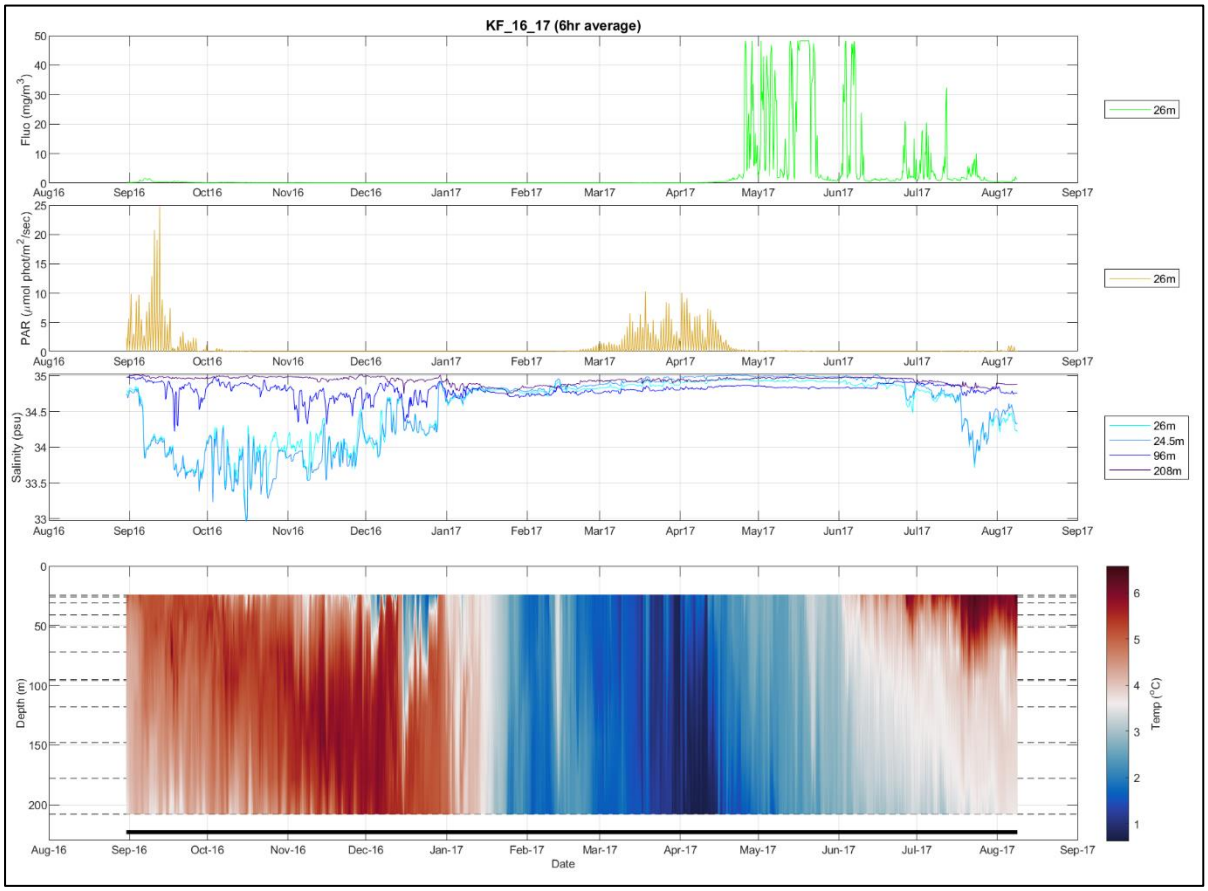

Figure 6: example of plot for step 8 – main plot

If a Hydrocat was installed on the mooring an extra timeseries plot will be created to show the additional variables:

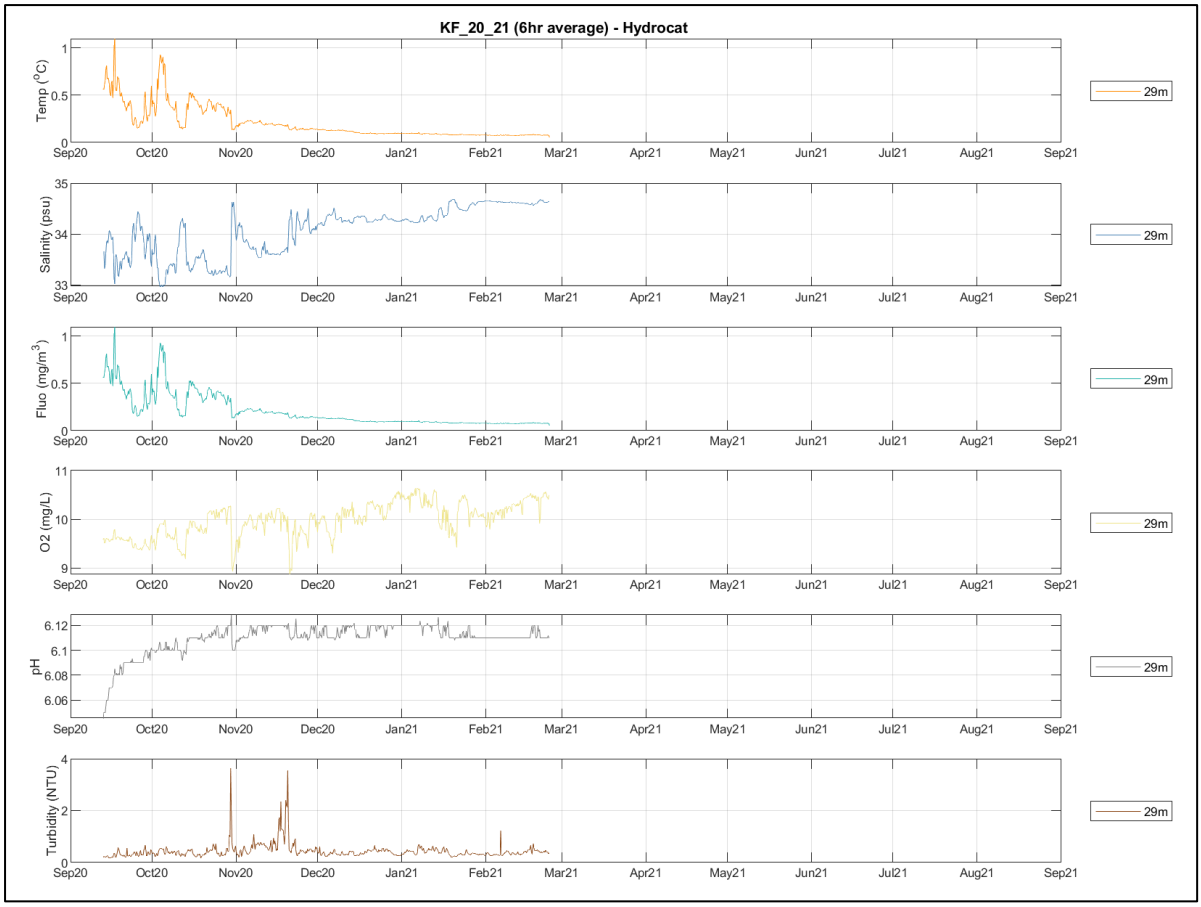

Figure 7: example of plot for step 8 – Hydrocat plot

3.2.10. Export data to csv

Script: KROP_export_csv.m
Sub-routines called: none
Input(s): MM_YY_YY_info.mat, MM_YY_YY_CTD.mat
Output(s): MM_YY_YY_raw_*.csv, MM_YY_YY_pro_*.csv, MM_YY_YY_avg_24h_*.csv, MM_YY_YY_grid_temp.csv

For each instrument the script exports the following data in standardised csv formats:

- Raw: from the RAW structure. This is essentially the exact same data as in the original files offloaded from the instruments, in a standardised format.
- Pro: from the PRO structure. This is the full resolution dataset, fully processed (out-of-water data removed, offsets applied to CT variables, and CT variables despiked).
- Avg_24hr: from the PRO_avg_daily structure. This is the fully processed data averaged in24h intervals.
- Grid_temp: from the PRO_grid_temp structure. This is the processed temperature data for all instruments, averaged every 6 hours and linearly interpolated every metre.

All exported files contain one header line and are comma separated.

The files will include the following variables (in this order):

Table 9: csv output file format

Variable	Description	Units / format
datetime	UTC time	DD-MMM-YYYY hh:mm:ss
nominal_depth	Nominal instrument depth from mooring design sheet	m
scan	Instrument scan number	n/a
jday	Julian day (1 = 1st January of the deployment year)	n/a
Optional (if the sensor is present on the instrument):		
pres	Recorded pressure	db
depth	Recorded depth	m
temp	Temperature (ITS-90)	°C
cond	Conductivity	mS/cm
Sal	Practical salinity	psu
sigma_theta	Density sigma-theta σ_θ	kg/m^3 - 1000
flc_V	Raw fluorometer readings	volts
flc	Chlorophyll-a concentration	µg/l
par_V	Raw PAR sensors readings	volts
par	PAR	µmol photons/m^2/s or µEinsteins / m^2 / s
tur_V	Raw turbidity sensor readings	volts
tur	Turbidity	FTU or NTU
oxy	Dissolved oxygen	mg/l
oxy_sat	Oxygen saturation	%
ph	pH	pH

3.2.11. Export data to NetCDF

Script: KROP_export_nc.m, KROP_export_nc_avg.m
Sub-routines called: none
Input(s): MM_YY_YY_info.mat, MM_YY_YY_CTD.mat, KROP_data_att.mat, KROP_data_att_avg.mat
Output(s): MM_YY_YY_pro_*.nc, MM_YY_YY_pro_avg_24h_*.nc

Those scripts export the fully processed data, in full resolution and 24h-averaged versions, into NetCDF format.

The data come from the relevant mat structures.

The metadata variables (e.g. mooring latitude, longitude, water depth, deployment date, etc) come from the info file.

The file attributes come from various sources:

- The default variable attributes (e.g. standard names, units, default processing level, etc) are stored in the KROP_data_att.mat and KROP_data_att_avg.mat files
- Some global file attributes are set in the script. These are attributes applicable to the whole mooring, which are unlikely to change (e.g. data type, featureType, area, etc).

- Some global file attributes are set in the mooring info file. These are attributes applicable to the full mooring, but which could change from year to year (e.g. contributors list, general comment about this specific mooring, etc).
- While the script is being run it gives the user the option to edit some of the individual variables attributes (processing level, QC indicator, comment) for each instrument.

All variables and units are described in the files.

Note: this script is the first step of the NetCDF file generation. A separate code has been developed by the University of Tromsø to amend and add further attributes to these files. In the future this step might be removed and the full NetCDF file generation carried out via the UiT tool.

4. Processing: ADCP data

4.1. Background & metadata

The ADCP data processing is done in Matlab, via a set of GUI interfaces. Those include some from the SAMS ADCP Backscatter GUI toolbox, as well as some developed specifically for KROP. The SAMS ADCP Backscatter GUI toolbox was developed by Finlo Cottier, Estelle Dumont, Laura Hobbs, Kathleen Johst and Jamie Rodgers; and uses functions written by Rich Pawlowicz and Mags Wallace.

Some of the GUIs require the user to specify input and output filenames. The filenames indicated in grey below are the recommended ones for KROP datasets (standard file naming conventions).

The metadata required for the ADCP processing (mooring location, water depth, instrument depth, orientation, deployment dates) is to be entered by the user in the different GUIs throughout the processing. It would be possible to add or set this information in the KROP_info_file used for CTD processing in future versions. This has not been done here, to avoid extensive rewrite of the existing SAMS code, and also to allow users to process the ADCP data independently of the moored CTD data (i.e.: the ADCP toolbox will work without the info_file).

4.2. Raw data download and formatting

- Download data as binary file (.000) in RDI WinSC.
- The downloaded data can sometimes be split into two files. In order to run the processing toolbox the raw data must be in one single file. To join two raw files use the dos command: **copy *filename1.000*/B + *filename1.001*/B *filename2*.000**
- Rename the file following the KROP naming convention:
  - **MM_YY_YY**_ADCP_**U**.000, or
  - **MM_YY_YY**_ADCP_**D**.000
- Move the file under **..\MM_YY_YY\bin\ADCP\1_raw**

4.3. Matlab toolbox processing

4.3.1. Read data

Script:	rdr_gui.m & .fig
Sub-routines called:	rdradcp.m
Input(s):	*MM_YY_YY*_ADCP_*O*.000
Output(s):	*MM_YY_YY*_ADCP_*O*_rdr.mat

This GUI is from the SAMS ADCP Backscatter toolbox.

It converts the raw binary RDI files (.000) to a mat structure, extracting the relevant configuration and recorded data. It uses the function rdradcp written by R. Pawlowicz (Oct 2006 version).

To run enter the command in Matlab:
`rdr_gui`

The GUI window will open (note: not all options will be visible from the start; they will appear after the previous step has been completed).

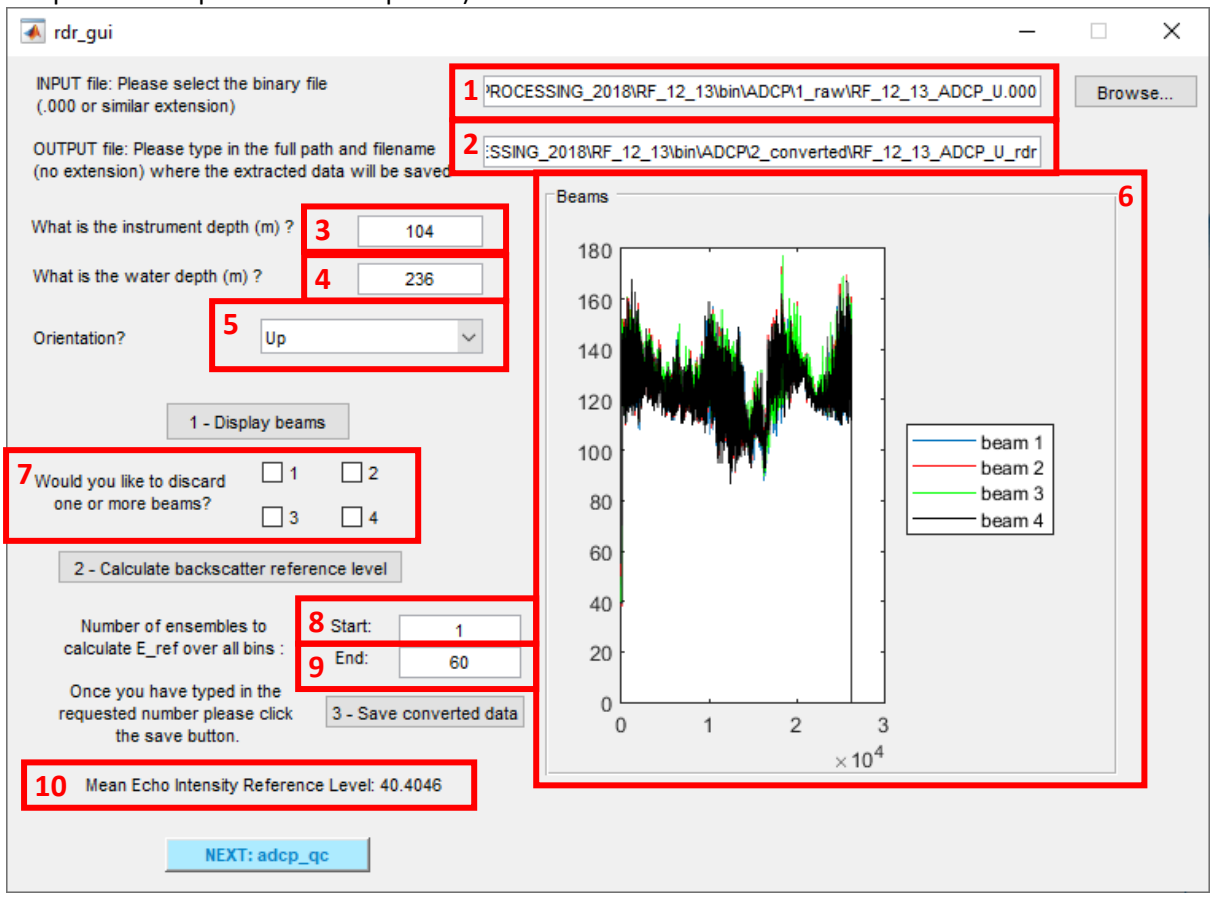

Figure 8: rdr_gui interface

- Browse for the input file (or type the full path and filename with extension) in box 1. This should be the raw datafile offloaded from the ADCP (extension .000). Datafiles split in two (or more) blocks should be merged prior to running this GUI (see instructions in Section 3.4.6).
- Type in the output path and filename without extension in the second window (or use copy and paste) in box 2.
- Type in the instrument depth in box 3 and the total water depth in box 4.
- Select ADCP orientation in drop-down box 5. Note: this parameter should be present in the file itself, but has been added here in case of faulty tilt sensor.
- Click on "1- Display beams". The script will then run for several minutes (depending on the file size and computer speed) to convert the binary data. Once done the echo intensity for all four beams will be plotted in the GUI (panel 6), allowing the user to select any faulty beams in panel 7 (for which the data will then be discarded). If all the beams appear normal (i.e. values within expected range and consistent across all beams) no beam should be selected.
- After selecting any eventual bad beams, press "2- Calculate backscatter reference level". A plot of Bin#1 echo intensity will open. Zoom in to identify periods when the ADCP is out of the water and note down the start and end ensemble numbers. Usually this is done using the pre-deployment

readings at the start of the record; if for some reason those are not available (e.g. if the ADCP started recording after deployment) then the readings post-recovery can be used. Those out-of-water readings will be much lower than the actual data and should be clearly identifiable.

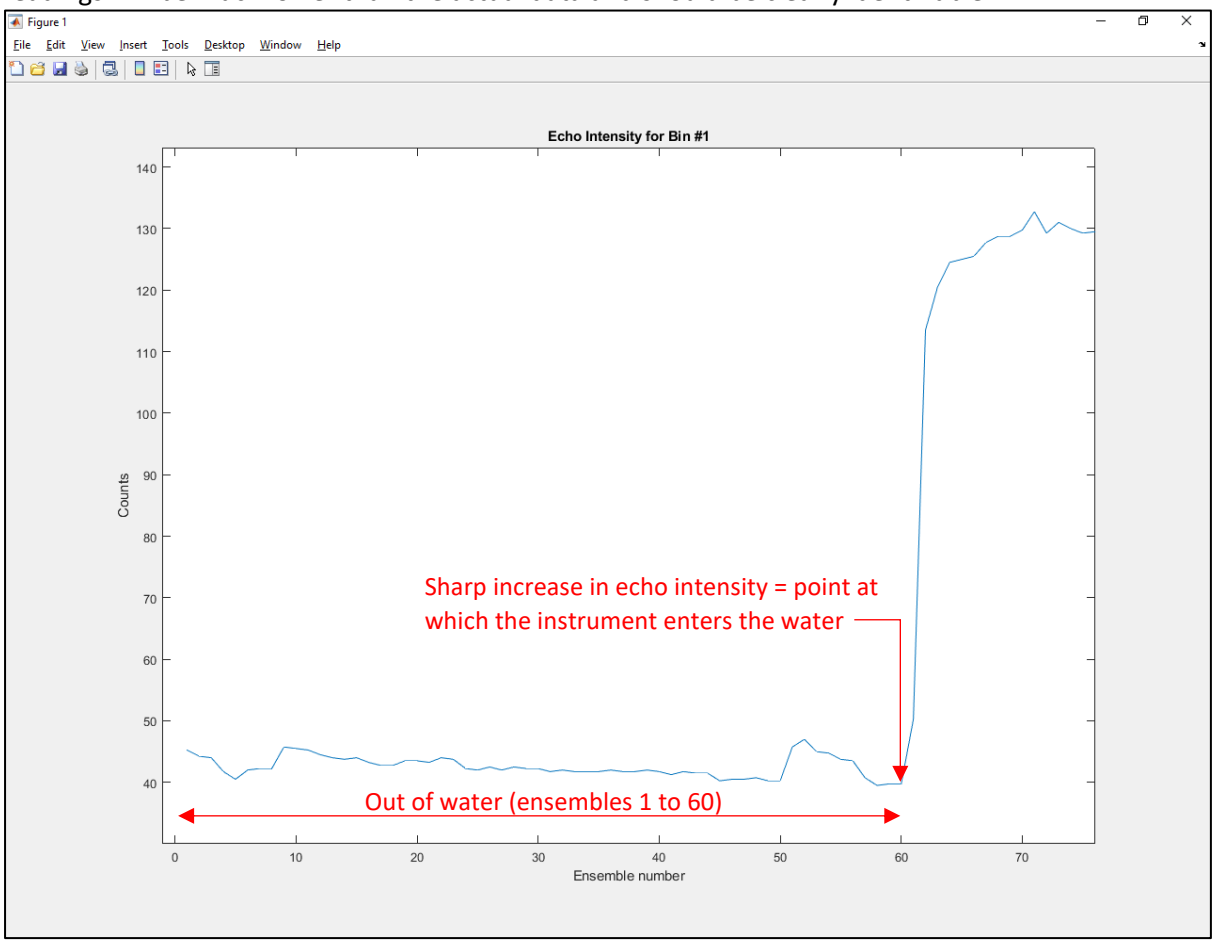

Figure 9: rdr_gui echo intensity plot example

- Enter the start and end ensemble numbers in boxes 8 and 9, and click on '3- Save converted data'. The script will then calculate the echo intensity for all beams over the selected ensembles and use those as the reference / background values for the calculation of backscatter later on. The average reference value (across all four beams) will be shown in panel 10.
- Click on the NEXT button to launch the QC GUI.

4.3.2. Crop & QC data

Script: adcp_qc_gui.m & .fig
Sub-routines called: adcp_qc.m
Input(s): *MM_YY_YY*_ADCP_*O*_rdr.mat
Output(s): *MM_YY_YY*_ADCP_*O*_rdr_qc.mat

This GUI is from the SAMS ADCP Backscatter toolbox.
It will add latitude and longitude information to the file, crop the raw data to the times specified by the user, perform a range of quality-checks, and save the data in either Matlab or Excel format.

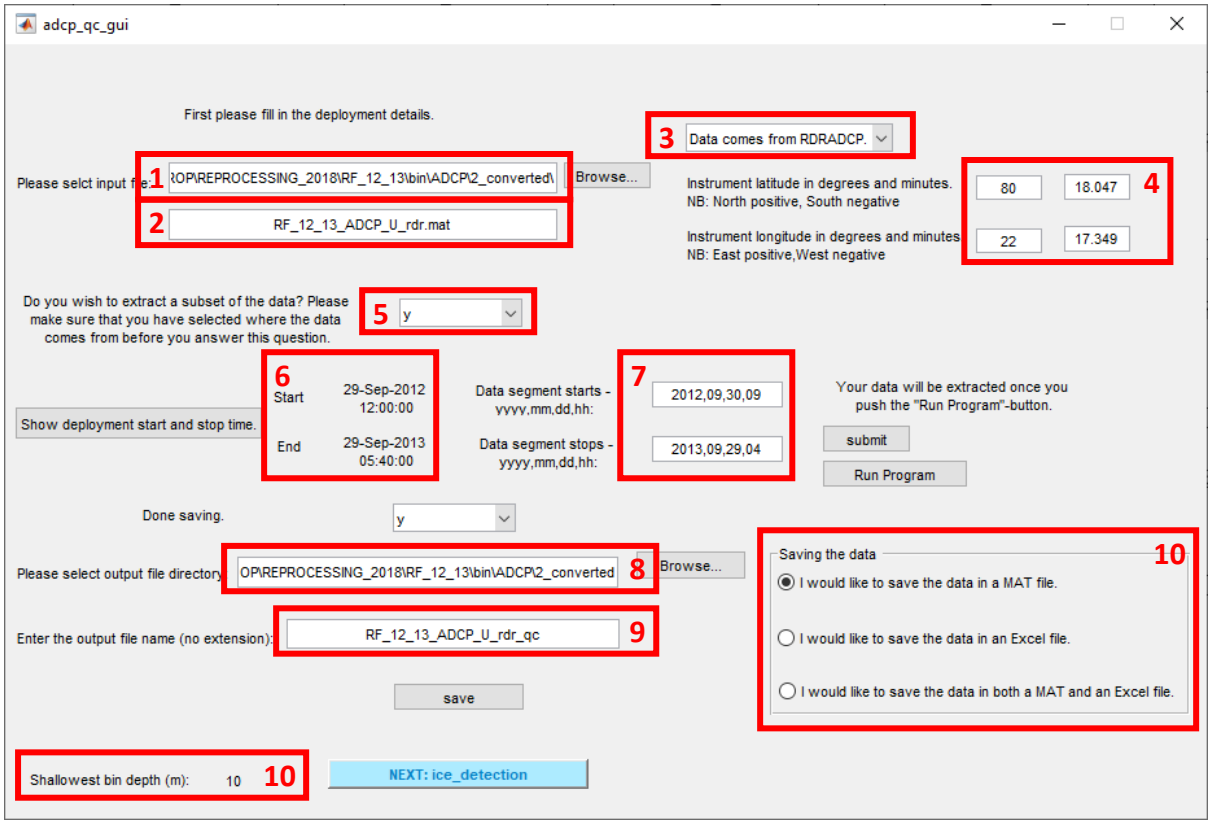

Figure 10: adcp_qc_gui interface

- Browse for the input file, or type the full path in box 1 and filename + extension in box 2. This should be the datafile obtained after running rdr_gui (or from WinADCP).
- In the drop-down menu (box 3) select the data origin, i.e. if the data has been processed using WinADCP or rdrADCP. If you select WinADCP the GUI will then ask you for the depth of the ADCP.
- In panel 4, input the latitude and longitude of deployment (info available from mooring design spreadsheet). Make sure you enter degrees in the first box and decimal minutes in the second, and that you follow the +/- coordinate conventions (North and East positive, South and West negative).
- Once the data origin has been selected, the next section will become available, allowing you to crop the dataset to mooring deployment and recovery times.
    - To keep the whole dataset, answer no ('n') in box 5 and click the "Run Program" button.
    - To crop the dataset, answer yes ('y') in box 5. By clicking on the "Show deployment start and stop time" you can see the total timespan of the data record (panel 6). Choose the start and end date of the period you wish to extract in panel 7. Dates must be in the format yyyy,mm,dd,hh. Click on 'submit' to check the times specified are within the dataset timestamps, then click on 'Run Program'.

Four QC plots will then be generated (figure 10). These should show a low scatter in the data, except for the last bin when close to the surface or seafloor.
The percentage return (4th figure) should be above 90% for most of the bins, although it can drop for the ones furthest from the transducer head. The last bin might show an increase when located at the surface or seafloor.

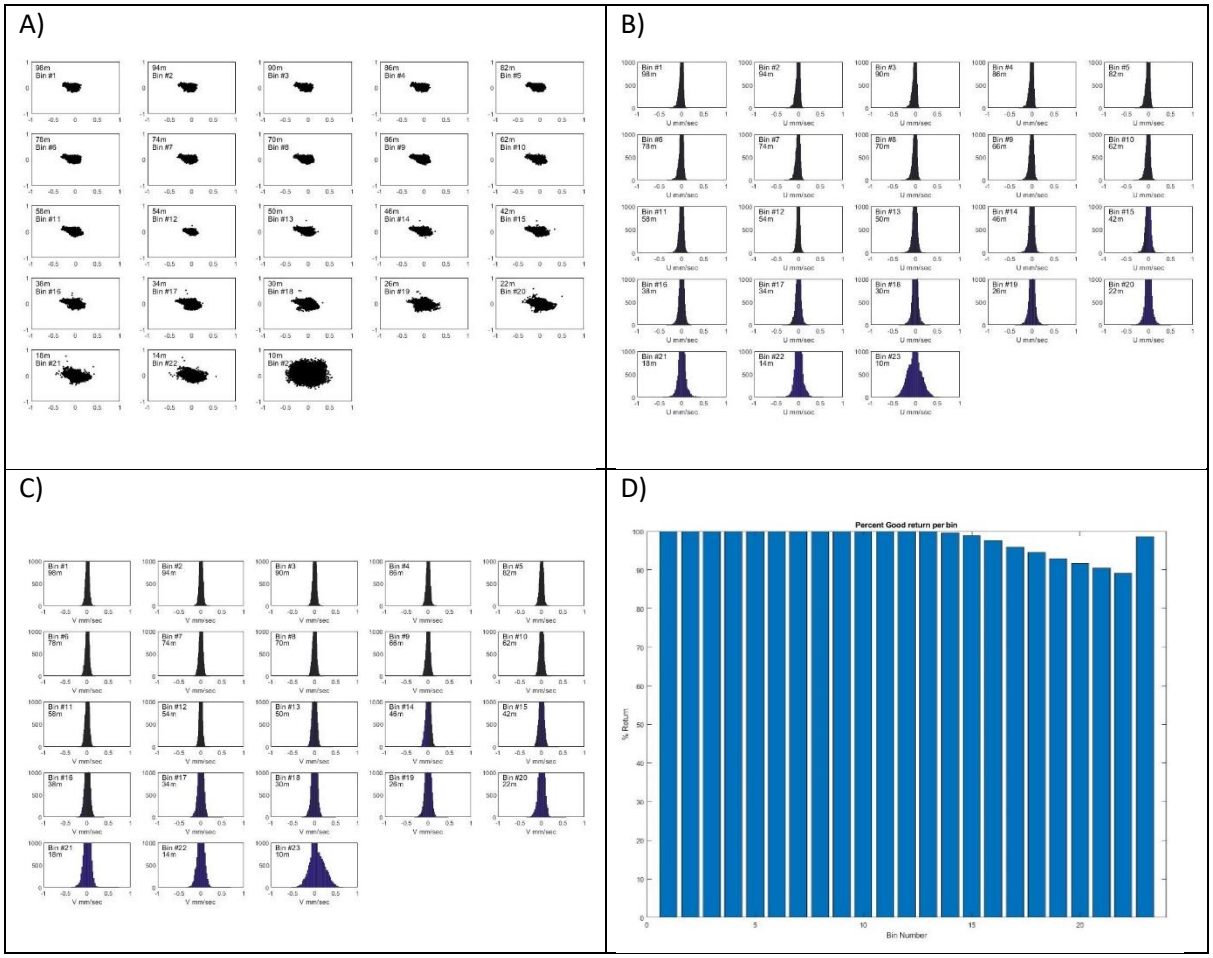

Figure 11: adcp_qc_gui QC plot examples. A) u and v scatter plots, B) u histogram, C) v histogram and D) percentage good return.

- To save the data, browse for or type in output directory (box 8) and enter output filename without extension (box 9). In panel 10 you can select in which format the data will be saved: mat and/or xls. Note: for the KROP processing we only require mat versions. Click on 'Save'.
- When done click on the NEXT button to launch the ice detection GUI. The shallowest bin depth will be shown in panel 10, allowing the user to determine if it is useable for ice detection (i.e. close enough to the surface). All datafiles must go through the ice detection GUI, for variable consistency across datafiles, even if the top bin did not reach the surface.

4.3.3. Ice detection

Script:	ice_detection_gui.m & .fig
Sub-routines called:	detect_ice.m
Input(s):	*MM_YY_YY*_ADCP_*O*_rdr_qc.mat
Output(s):	*MM_YY_YY*_ADCP_*O*_rdr_qc_ice.mat, Ice_detection_*MM_YY_YY*_ADCP_*O* _rdr_qc.jpg

This GUI is from the SAMS ADCP Backscatter toolbox.
It allows to visually determine ice cover periods by selecting periods of low horizontal speed variance in the surface bin (see Hyatt et al., 2008). The GUI uses the function detect_ice written by M. Wallace (May 2008).

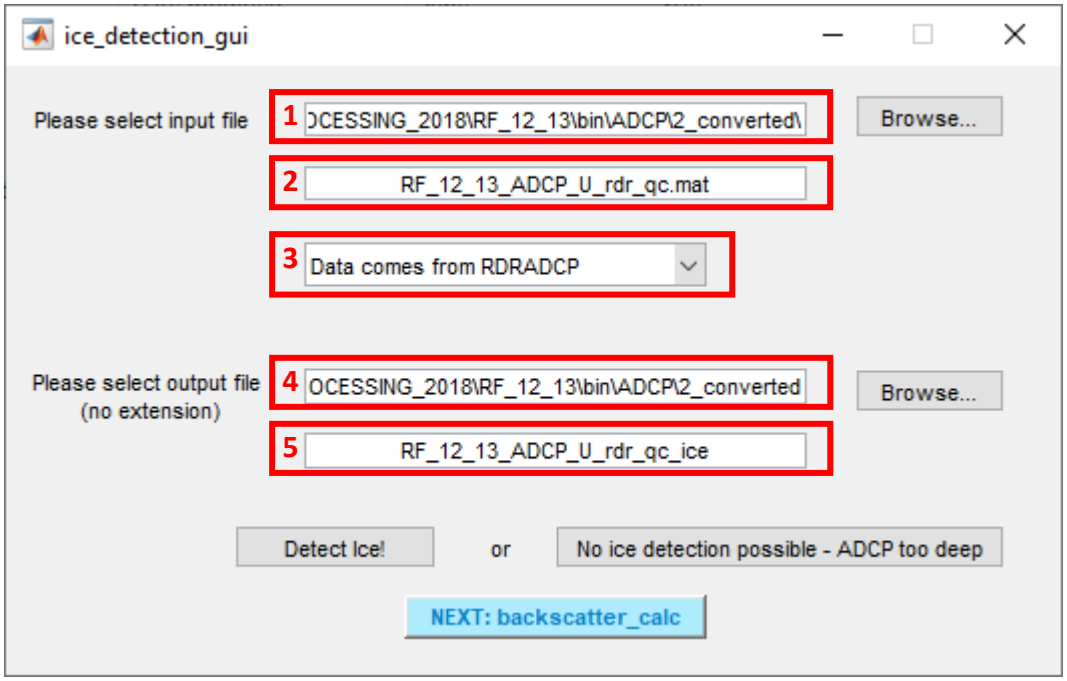

Figure 12: ice_detection_gui interface

- Browse for the input file, or type the full path in box 1 and filename + extension in box 2. This should be the datafile obtained after running adcp_qc_gui.
- In the drop-down menu (box 3) select the data origin, i.e. if the data has been processed using WinADCP or rdradcp.
- Browse for or type in output directory in box 4 and filename without extension in box 5.
- If the ADCP data reaches the surface (see 'Shallowest bin depth' on previous GUI), press 'Detect Ice'. A figure will open showing the horizontal velocity in the top bin (figure 12). Follow the instructions in the command window to select the ice-covered periods. For information on indicators of ice presence see Hyatt et al. (2008). Several periods can be selected. If no ice period can be identified, simply press 'Enter' when asked to select the start and end point.

When done, the variable 'ice_cover' will be added to each data record (0 = no ice, 1=ice), along with the bin number used for the ice detection. The figure will also be saved.

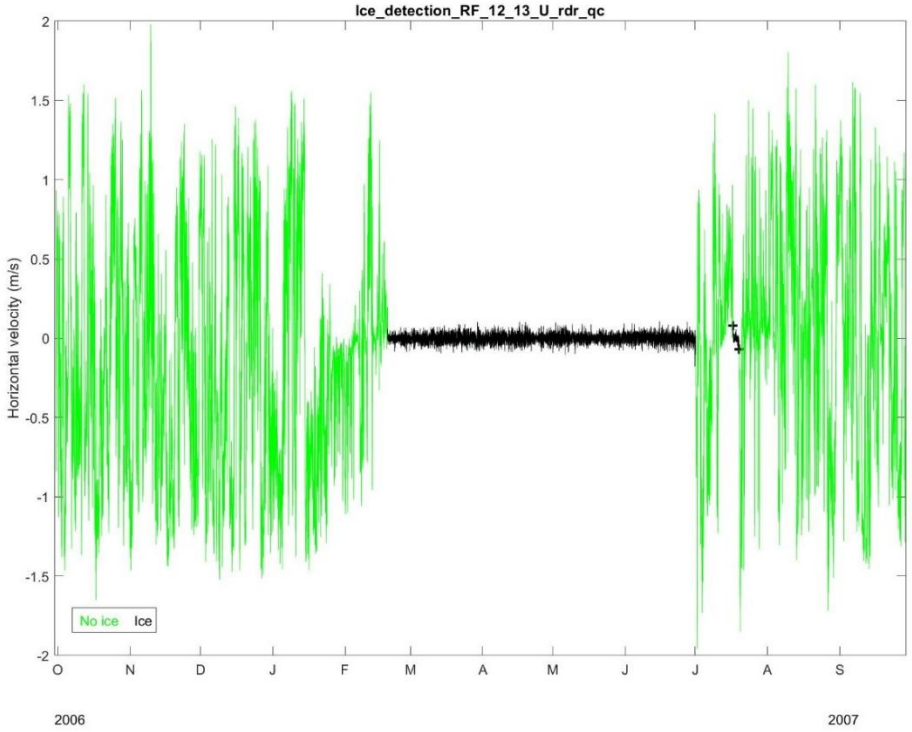

Figure 13: final ice detection figure example, showing the horizontal velocity in the top (surface) bin. Black periods have been selected by the processor as ice-covered.

- If ice detection is not possible (e.g. ADCP too deep, or looking downwards), press 'No ice detection possible'. The variable 'ice_cover' will be added to each data record (value of -1), and ice_bin will be set to NaN.
- When done click on the NEXT button to launch the backscatter calculation GUI.

4.3.4. Backscatter calculation

Script: backscatter_cal_gui.m & .fig
Sub-routines called: Sv_calc.m
Input(s): *MM_YY_YY*_ADCP_*O*_rdr_qc_ice.mat
Output(s): *MM_YY_YY*_ADCP_*O*_rdr_qc_ice_sv.mat

This GUI calculates acoustic backscatter (Sv) using the function Sv_calc.m written by F. Cottier (July 2003) and updated by M. Wallace (May 2008), based on Deines, 1999.

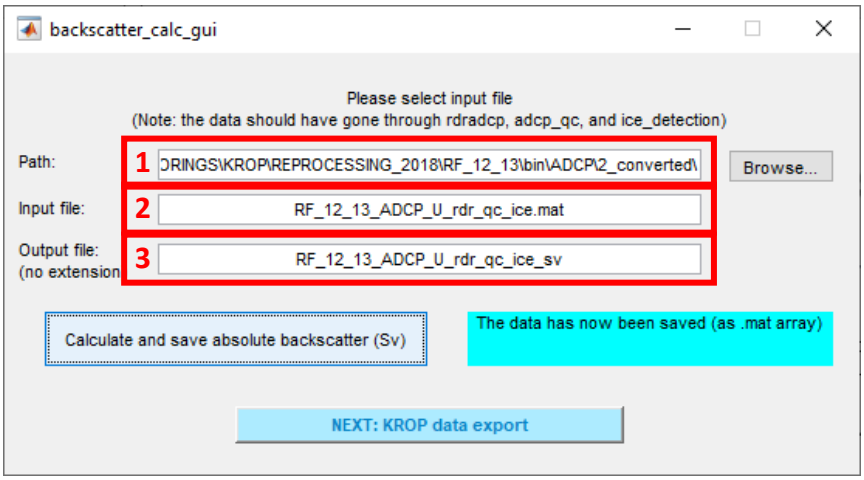

Figure 14: backscatter_calc_gui interface

- Select directory, input file and output file in boxes 1, 2 and 3. Press 'Calculate and save absolute backscatter'.
Sv will be calculated for each data record and saved in the output file (along with all other existing variables). Three control plots will be generated, for reference (figure 14).

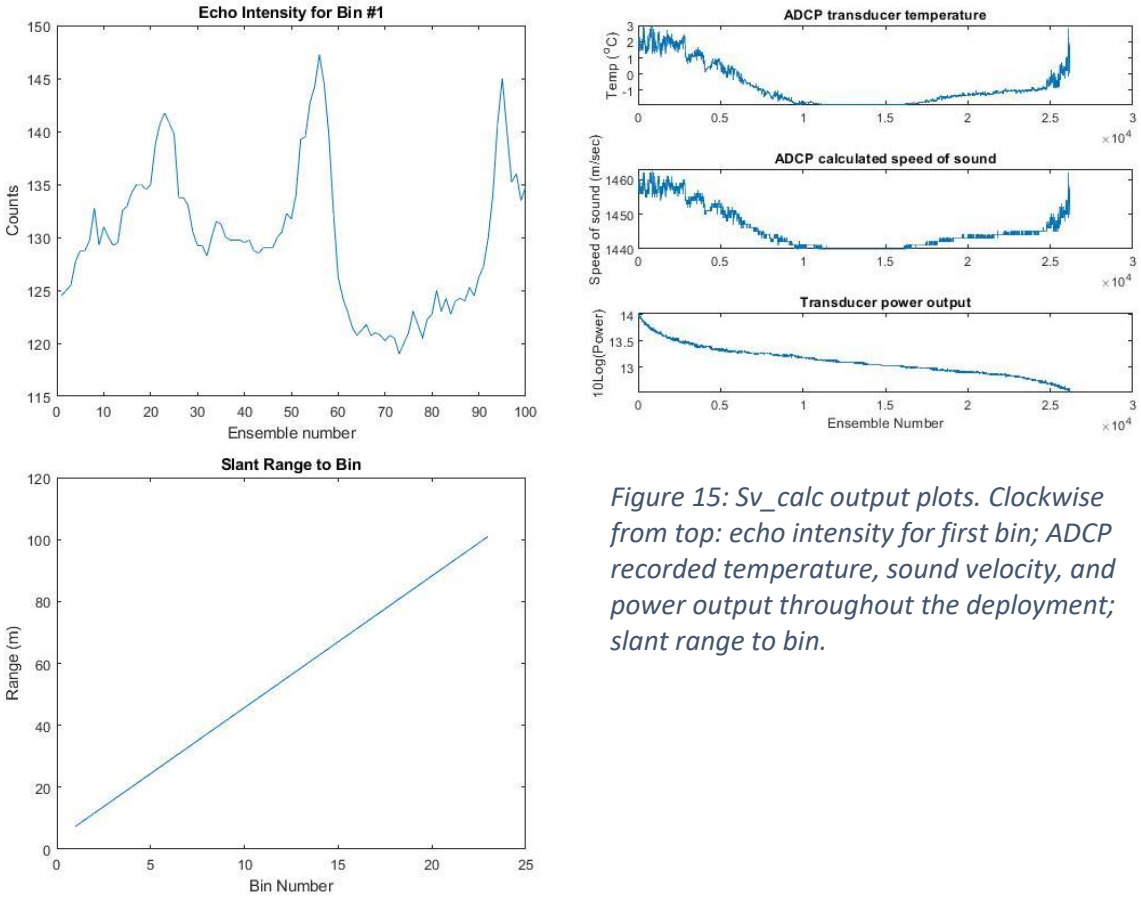

Figure 15: Sv_calc output plots. Clockwise from top: echo intensity for first bin; ADCP recorded temperature, sound velocity, and power output throughout the deployment; slant range to bin.

- When done click on the NEXT button to launch the KROP export & plot GUI.

4.3.5. KROP data export and plot

Script: krop_explort_plot_gui.m & .fig
Sub-routines called: KROP_export_adcp_mat.m, interpolate_adcp_data_30min_weekly.m, plot_adcp_data_weekly.m
Input(s): *MM_YY_YY*_ADCP_*O*_rdr_qc_ice_sv.mat
Output(s): …\mat*MM_YY_YY*_ADCP_*O*.mat
…\csv\ADCP*MM_YY_YY*_ADCP_*var_yyyy-mm-dd.csv*
…\plots\ADCP*MM_YY_YY*_ADCP_*varvar_yyyy-mm-dd.png*

This GUI has been developed specifically for exporting and plotting ADCP data for KROP. It consists of three separate functions:

- Export final data in standardised KROP mat format
- Export weekly csv grids for selected variables
- Plot weekly data for selected variables

First browse for or enter your KROP root directory in box 1 (i.e. the directory containing all the moorings sub-directory). Select mooring in drop-down list 2. Note: the GUI will add any subdirectories found in the specified root directory that loosely resemble KROP mooring names (those with two

underscores in the name). As a result, it might pick up subdirectories that do not actually contain mooring data, and might not recognise a mooring directory if it does not follow the KROP naming convention.

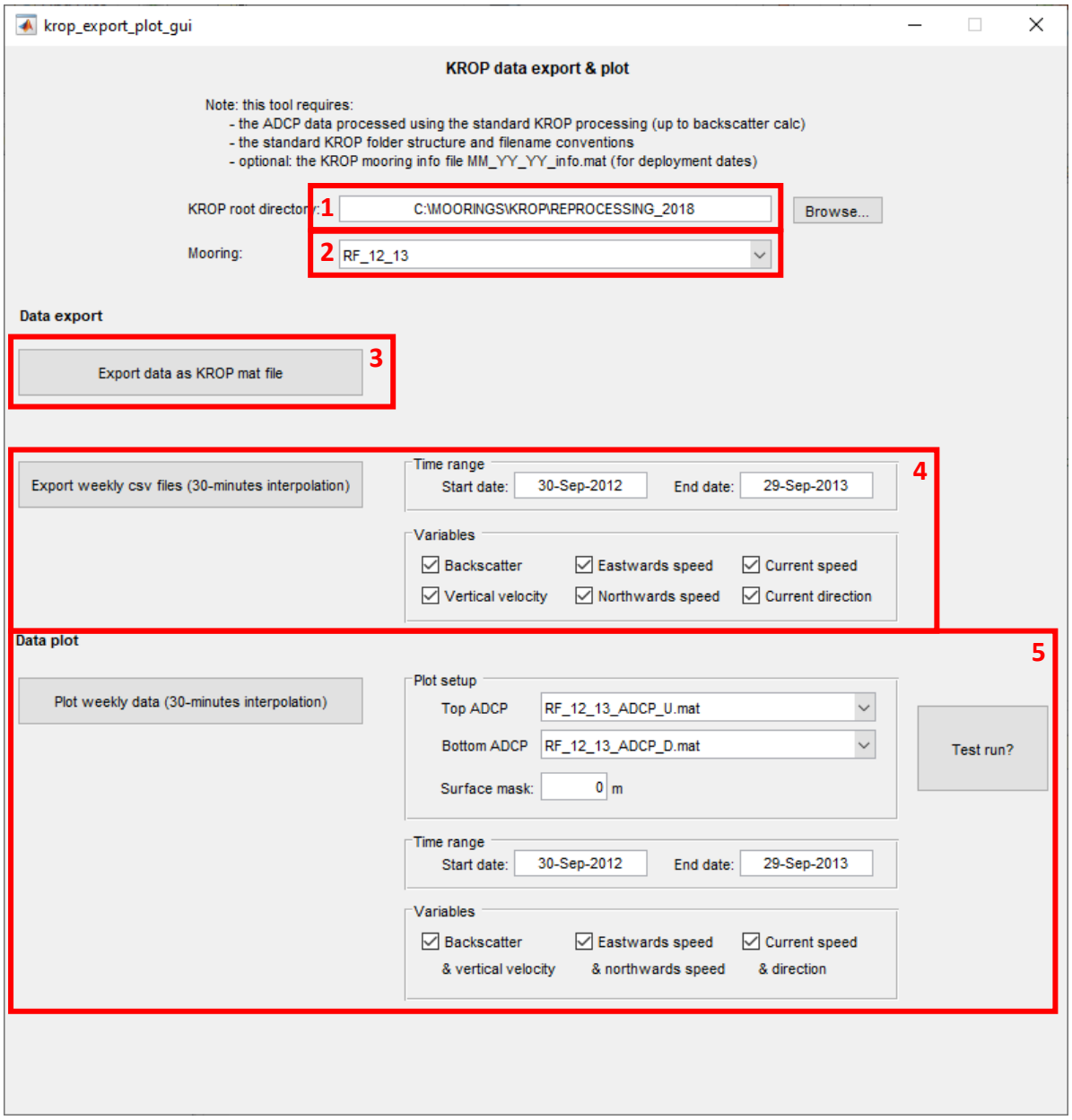

Figure 16: krop_export_plot_gui interface - part 1

4.3.5.1. Export KROP ADCP mat files

Script: KROP_export_adcp_mat.m
Input(s): …\bin\ADCP\2_converted*MM_YY_YY*_ADCP_*O*_rdr_qc_ice_sv.mat
Output(s): …\mat*MM_YY_YY*_ADCP_*O*.mat

Click on the "Export […]" button (3). The script will look for any datafiles matching the expected nomenclature (*MM_YY_YY*_ADCP_*_sv.mat) in the …\bin\ADCP\2_converted\ directory, extract a subset of variables (Table 10) and export those in a mat format.

Table 10: list of variables included in KROP ADCP mat

Variable	Description	Units	Dimensions
time	Epoch time	Seconds since 01-Jan-1970 00:00:00 UTC	-
mtime	Matlab time	Days since 00-Jan-0000	-
latitude	Mooring latitude	Decimal degrees (WGS84)	1
longitude	Mooring longitude	Decimal degrees (WGS84)	1
seafloor_depth	Mooring seafloor depth	m	1
nominal_depth	Instrument nominal depth	m	1 x time
pressure	Instrument recorded pressure	dbar x1000	1 x time
bin_depth	Depth of measurement vertical bins (middle)	m	-
ice_cover	Ice cover	0 = no ice, 1 = ice	1 x time
pitch	Instrument pitch	degrees	1 x time
roll	Instrument roll	degrees	1 x time
u	Eastwards velocity	m/s	time x bin_depth
v	Northwards velocity	m/s	time x bin_depth
w	Vertical velocity	m/s	time x bin_depth
Sv	Backscatter	dB	time x bin_depth
velocity_err	Velocity error	m/s	time x bin_depth

Those mat files can then be converted to NetCDF using the tool developed by UiT.

4.3.5.2. *Export KROP weekly ADCP csv files*

Script: interpolate_adcp_data_30min_weekly.m
Input(s): …\mat*MM_YY_YY*_ADCP_*O*.mat
Output(s): …\csv\ADCP*MM_YY_YY*_ADCP_*var*_*yyyy-mm-dd*.*csv*

var (csv) = sv [backscatter], w [vertical velocity], u [eastwards velocity], v [northwards velocity], hv [overall horizontal velocity / current speed], or dir [current direction]
yyyy-mm-dd = first day of the week exported / plotted

In panel 4, select:

- Start and end date of data to export. The default values will be the start and end time for the full mooring deployment.
- Variables to export

Press the "Export […]" button. For each sleected variable, the data will be interpolated every 30 minutes (on the hour and at 30 minutes past) for each bin, and saved as weekly grids in csv format. The first row of each file contains the bin depth, and the first column the time reference (in epoch format).

4.3.5.3. *Plot weekly ADCP data*

Script: interpolate_adcp_data_30min_weekly.m, plot_adcp_data_weekly.m
Input(s): …\mat*MM_YY_YY*_ADCP_*O*.mat
Output(s): …\plots\ADCP*MM_YY_YY*_ADCP_*varvar*_*yyyy-mm-dd*.*png*

varvar = svw, uv, or currents (hv + dir)
yyyy-mm-dd = first day of the week exported / plotted

In panel 5, first select the vertical order of the ADCPs on the mooring line.

The “surface mask” box allows the user to set a distance from the surface to not show on the plots (generally the bin hitting the surface can be noisy). In order to determine the depth of the surface bin, leave “Surface mask” at 0m and click “Test run?”. This will generate a single figure of Sv and W on which the user can zoom in (Figure X). After determining the value the figure can be closed. Enter the value found as the surface mask (or leave at 0 if no masking is required).

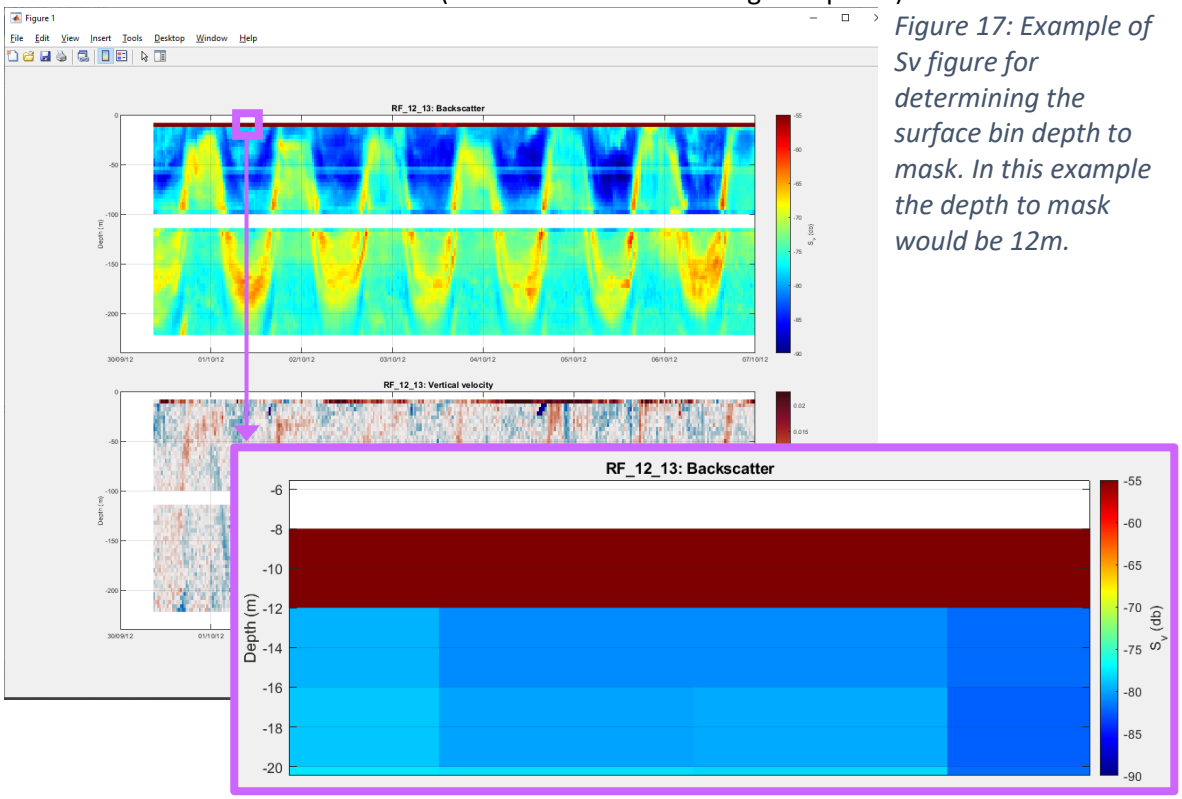

Figure 17: Example of Sv figure for determining the surface bin depth to mask. In this example the depth to mask would be 12m.

Then select start date, end date and variable pairs to plot. Click “Plot weekly data […]”. Plots will be generated and saved under …/plots/…ADCP.

5. References

K. L. Deines, *Backscatter estimation using Broadband acoustic Doppler current profilers*, Proceedings of the IEEE Sixth Working Conference on Current Measurement (Cat. No.99CH36331), 1999, pp. 249-253, DOI: 10.1109/CCM.1999.755249

J. Hyatt, M. Visbeck, R.C. Beardsley, W. Brechner Owens, *Estimating sea-ice coverage, draft, and velocity in Marguerite Bay (Antarctica) using a subsurface moored upward-looking acoustic Doppler current profiler (ADCP)*, Deep-Sea Research II 55, 2008, pp 351–364, DOI: 10.1016/j.dsr2.2007.11.004